Encryption Methods and Key Management Services for Secure Cloud Computing: A Review

Tristan L. Moore
Department of CSIT
Saint Cloud State University
St. Cloud, Minnesota, 56301
tlmoore@go.stcloudstate.edu

Samuel S. Conlon
Department of CSIT
Saint Cloud State University
St. Cloud, Minnesota, 56301
ssconlon@stcloudstate.edu

Anushka U. Hewarathna
Department of MSIA
Saint Cloud State University
St. Cloud, Minnesota, 56301
auhewarathna@go.stcloudstate.edu

Thivanka B. M. Dissanayaka M.
Department of MSIA
Saint Cloud State University
St. Cloud, Minnesota, 56301
thivankabm@gmail.com

Akalanka B. Mailewa
Department of CSIT
Saint Cloud State University
St. Cloud, Minnesota, 56301
amailewa@stcloudstate.edu

Abstract

Utilization of public Cloud Service Providers (CSPs) has increased drastically since its inception, with many businesses using a Software-as-a-Service (SaaS) business model, meaning their entire business is run on the cloud. Due to this business model's increase in popularity in recent years, CSPs need to make security of data in the cloud their top priority and give their customers the proper tools they need to protect their data while using their services. This paper intends to give a comprehensive overview of the Encryption Key Management Services offered by two of the most frequently use CSPs: Amazon Web Services (AWS) and Google Cloud Platform (GCP). In this research, AWS Key Management Service (KMS) and Google Cloud Key Management Service offerings were tested hands-on within each respective CSPs cloud console. Each Key Management Service was thoroughly tested to fully understand their capabilities, use cases, and faults. Based on the findings it can be observed that AES-256 was the clear winner for symmetric encryption key use and RSA for asymmetric encryption keys. In addition, this paper identifies and presents several open research problems in the field of cloud-based encryption as well.

Keywords: Risks; Privacy; Performance; Amazon Web Services (AWS); Encryption; Google Cloud Platform (GCP); Key Management Service; Could Security

1 INTRODUCTION

This paper reviews the encryption methods and key management services for secure cloud computing by using AWS and GCP. This section briefly overview the context and definitions that will give guidance to the overall subject of this discussion.

1.1 Cloud Service Provider (CSP)

A cloud service provider (CSP), is a company that offers some components of cloud computing [1] [2]. The most common methods to leverage CSP services are Infrastructure as a Service (IaaS), where a Company maintains on premises servers and datacenters, but pushes some of the workload and storage to the cloud. Platform as a Service (PaaS), where a Company maintains their own application engine, but leverages the cloud to deploy their application, and Software as a Service (SaaS) where a Company completely utilizes the cloud for hosting their application and storage. The most widely used public cloud service providers include Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure [3][4].

1.2 Symmetric Encryption

In symmetric encryption, only one key is used to encrypt and decrypt data [5] this is typically used for encrypting storage volumes, databases, and storage buckets as the computing overhead is generally much lower in symmetric encryption than asymmetric. The industry standard symmetric encryption algorithm utilized by all public CSPs is Advanced Encryption Standard with 256-bit length key (AES-256) [6].

1.3 Asymmetric Encryption

Asymmetric encryption, also known as public key cryptography, encrypts and decrypts data using two separate yet mathematically connected cryptographic keys. These keys are known as a “public key” and “private key.” Together, they’re called a “public and private key pair.” [5][7] Asymmetric encryption is typically used in end-to-end encryption between a client and server. On a website for example, the owner of the website maintains the private key with a certificate signed by a certified authority (CA) and distributes a public key to each client whenever they visit the website [7][8].

1.4 Key Management Service

The three most widely used CSPs (AWS, GCP, and Azure) all offer key management services [9]. A key management service allows customers to centrally manage all of their encryption keys within the cloud. Customers can create, manage access, delete, and rotate their encryption keys within their key management service dashboard. This is particularly useful as the service is integrated with Identity and Access Management (IAM) services, allowing for granular distribution and restriction of access to these encryption keys. These encryption keys can encrypt access to storage volumes, databases, storage buckets, and even different cloud-based services such as infrastructure monitoring logs [10].

2 BACKGROUND

Security is generally a large concern among previous works regarding the topic of key management services in the cloud, this is due to the fact that a single point of failure is created when customers become over-reliant on cloud services to manage resources and sensitive customer data [11]. Depending on the type of data stored within databases, storage volumes, and storage buckets in the cloud, customers want a full guarantee that their data will be safe utilizing the CSPs methods of encryption key management [12]. To assure the security in cloud based environments, this paper presents two frameworks as follows as prior work.

Ahmad, Shahnawaz, et al. [13], the authors go in depth about secure encryption key creation and processes that must be in place to ensure confidentiality, integrity, and availability are not compromised for these keys which are managed through a CSP. Random bit generation for keys must be truly random enough to avoid the key encryption algorithm from being compromised and keys being distributed must be protected during transmission. The authors then proposes a more generic solution to stop data loss by implementing a Cloud-Based Data Loss Prevention (DLP) framework. The proposed framework consists of six major categories:

1. Data Discovery and Classification – This category focuses on determining what data is worth protecting, as well as associating a level of protection for each document. For example, levels of sensitivity could vary from public information which requires no protection, to confidential information which must be stored encrypted at rest [13][14].
2. Advanced OCR & NLP Capability – This is the process of utilizing machine learning and artificial intelligence software to determine what data is stored in documents and how sensitive the information is. This is important as an IT security department doesn't necessarily have the time to sift through all stored documents and classify them based on sensitivity of the data [13][15].
3. Fingerprinting & Tagging – This is the process of identifying IT assets, this can be managed using an asset management platform or mobile device management (MDM) tool. This process is important to identify mission-critical information assets and what levels of protection each asset may require to keep the business operational [13][16].
4. Clipboard Monitoring – This process involves logging all user interactions within the cloud-based environment. This is important for identifying potential disgruntled employees and insider threats, as well as unusual/malicious activity occurring in the environment [13][17].
5. Content-Based Policy & Rules – This process involves establishing baselines that all employees must adhere to, as well as give direction to restoring the environment in the event of a disaster or disruption. Other policies could include ethical behavior requirements for employees and an incident response plan in the event of a potential data breach [13][18].

6. Compliance Management – This process involves adhering to various compliance frameworks and requirements to assure user entities of the business that their data is being properly protected [13][19].

This framework establishes defense-in-depth by hardening security at all layers and facets a business may operate on.

Noor, Talal, et al. [20] the authors propose a "Trust Management Framework" for public cloud service providers (e.g., AWS, GCP, and Azure) to maintain a healthy trust relationship between the customer and CSP. This framework proposes three layers:

1. Trust Feedback Sharing Layer – This layer consists of different parties including cloud service consumers and providers, which give trust feedback to each other. The feedback is maintained via a module called the Trust Feedback Collector. The feedback storage relies on the trust management systems, in the form of centralized, decentralized, or within the cloud environment through a trusted cloud service provider. Within the Trust Feedback Sharing Layer lies four dimensions [20]:

1.1. Credibility – This dimension refers to the quality of the information or service that makes cloud service consumers or provides trust the information or the service [20][21].

1.2. Privacy – This dimension refers to the degree of sensitive information disclosure cloud service consumers may fall victim to during interactions with the trust management system [20][22].

1.3. Personalization – This dimension refers to the degree of autonomy cloud service consumers and providers adhere to within the rules of the trust management system [20][23].

1.4. Integration – This dimension refers to the ability to integrate different trust management perspectives and techniques [20][24].

2. Trust Assessment Layer – This layer represents the core of any trust management system: trust assessment. The assessment might contain more than one metric. TAL handles a huge amount of trust assessment queries from several parties through a module called the Trust Result Distributor. This typically involves checking the trust results database and performing the assessment based on different trust management techniques. TAL delivers the trust results to a database in the trust results distribution layer through the module of the trust result distributor. This procedure is taken to avoid redundancy issues in trust assessment. Within the Trust Assessment Layer lies six dimensions [20]:

2.1. Perspective – This dimension depends on the focus of the trust management approach of the framework, some may focus on the consumer's perspective, while others may focus on the CSPs perspective [20][25].

2.2. Technique – This dimension refers to the degree to which a technique can be adopted by the trust management system to manage and assess trust feedback [20][26].

2.3. Adaptability – This dimension refers to how quickly the trust assessment function can adapt to the changes of involved parties such as the cloud service consumer and CSP [20][27].

2.4. Security – This dimension refers to the robustness of the trust assessment function against malicious behaviors and attacks. This dimension is very important as a cloud service consumer is trusting the CSP to offer services to allow for the cloud service consumer to protect their data, as well as deferring other aspects of security such as physical data center access to the CSP [20][28].

2.5. Scalability – This dimension refers to one of the most fundamental aspects of the cloud. One of the most appealing aspects of the cloud is that it operates on a pay-as-you-go model, and only pay for the compute resources used. The cloud is heavily relied upon to be able to scale for any amount of network demand and to continuously grow its capabilities as time goes on [20][29].

2.6. Applicability – This dimension refers to the degree that the trust assessment function can be adopted to support trust management systems deployed for cloud services [20][30].

3. Trust Results Distribution Layer – Similar to TFSL, this layer consists of different parties including cloud service consumers and providers, which issue trust assessment inquiries about other parties (e.g., a cloud service consumer inquiry about a specific cloud service). All trust assessment inquiries are transmitted to the trust assessment function through the module of trust assessment and results distributor. The final results are maintained in a database where cloud service consumers and providers can retrieve. Within the Trust Results Distribution Layer lies four dimensions [20]:

3.1. Response Time – This is the time that the trust management system requires to handle trust assessment inquiries, access feedback, and distribute trust results [20][31].

3.2. Redundancy – This dimension refers to the degree of redundancy support that the trust management system maintains in order to manage and assess the trust feedback [20][32].

3.3. Accuracy – This dimension refers to the degree of correctness of the distributed trust results that can be determined through one or more accuracy characteristics such as the unique identification of feedback and using the proper assessment function security level [20][33].

3.4. Security – This dimension refers to the degree of protection that the trust assessments and results distributor has against malicious behaviors and attacks [20][34].

3 METHODOLOGY

This research uses a hands-on approach for the analysis of each key management service available from each of the two largest cloud service providers (AWS and GCP). The authors have utilized demo accounts within each cloud service and tested the capabilities of each service. The following investigation will provide figures to guide each of the testing steps in this analysis.

3.1 AWS KEY MANAGEMENT SERVICE (KMS)

AWS KMS allows customer to create, manage, rotate, and delete their customer-owned encryption keys. Each encryption key managed within KMS includes attached metadata that customers can view, such as the key ID, key spec, key usage, creation date, description (optional), key state, and key material. Key spec refers to the type of encryption the key utilizes, this could be either symmetric or asymmetric, as well as the type of algorithm the key supports. By default, AWS KMS keys use AES-256 encryption for symmetric keys. Asymmetric key algorithms are typically customer defined, as asymmetric encryption is rarely utilized in AWS KMS [35]. AWS supports RSA and Elliptic Curve (ECC) asymmetric key pairs [36]. Key usage determines how the encryption key is used, a key can either be used to encrypt and decrypt in most cases. However, in the case of asymmetric encryption, a key can also be used for signing and verifying signatures. Each KMS key can only have one type of usage associated with them. Key state determines the current status of the key, this could be enabled, disabled, or pending deletion. Key material refers to the string of bits that make up the encryption algorithm of the key, this must be kept secret to protect the cryptographic operations that use it. However, public key material is designed to be shared [37].

3.1.1 AWS KMS – Envelope Encryption

When a customer encrypts their data with an encryption key, the data is protected. However, the key remains exposed. To solve this issue, envelope encryption can be used within KMS. This concept involves encrypting the encryption key that encrypts the data (referred to as the ‘data key’) with another encryption key (referred to as the ‘root key’) [38]. An AWS customer can import their plaintext data key into KMS, which will encrypt it with a root key, the root key can never leave the KMS module unencrypted. To use the key, it must be called within KMS. Envelope encryption also allows for the combination of multiple algorithms, a symmetric root key can be used to encrypt an asymmetric data key. Reference the figure below which further describes the concept of envelope encryption [39].

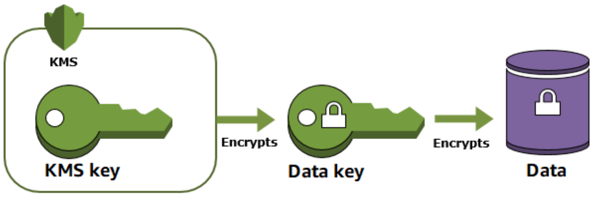


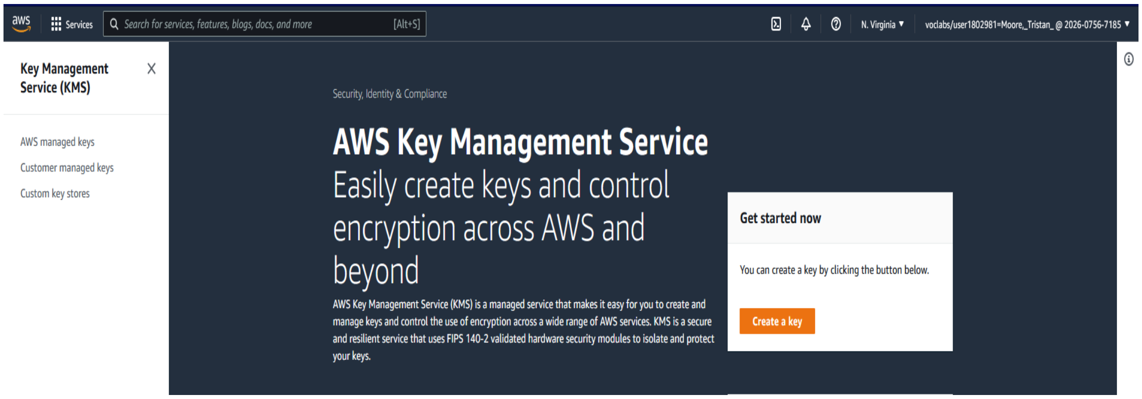
Figure 1: AWS KMS Envelope Encryption [39]

3.1.2 AWS vs Customer Managed Keys

Within KMS, there are two types of encryption key ownership, AWS managed, and customer managed. For customer managed keys, the customer retains full ownership of the key. The customer can enable, disable, rotate, and delete the key. AWS managed keys are KMS keys in a customer's account that are created, managed, and used on their behalf by an AWS service integrated with KMS top protect the customer's resources in the service. Some examples of services include AWS S3 (global storage buckets), AWS CloudTrail (infrastructure logging service), and AWS Inspector (vulnerability scanning service for containers and compute instances). Customers can still view their AWS managed key's policies and audit their use [39][40]. However, they cannot change the policies, rotate, or delete the keys. AWS managed keys are rotated on an annual basis.

3.1.3 AWS KMS Walk-Through Implementation

To further test the capabilities of AWS KMS, a demo account was created to test the service. AWS KMS was used to create a KMS key the figure below shows the AWS KMS dashboard.


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
voclabs/user1802981=Moore,_Tristan_@ 2026-0756-7185
Key Management Service (KMS)
AWS managed keys
Customer managed keys
Custom key stores
Security, Identity & Compliance
Get started now
Create a key

AWS Key Management Service

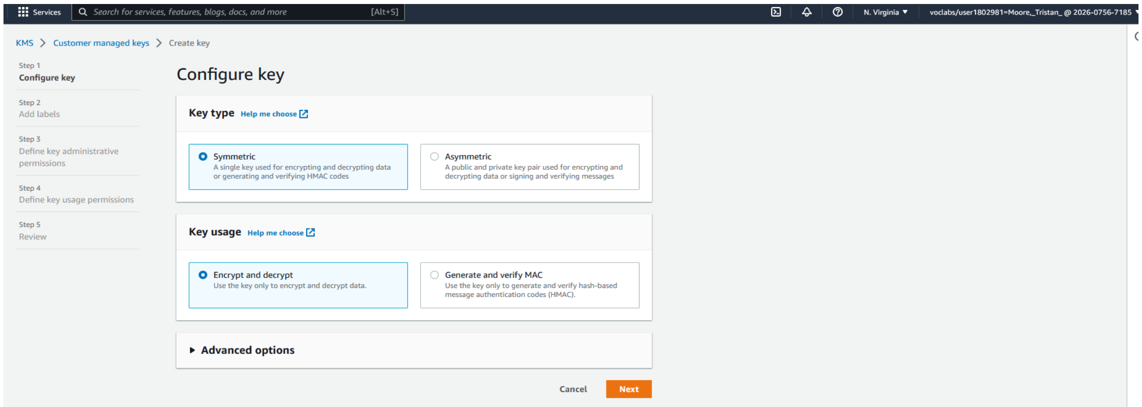
Easily create keys and control encryption across AWS and beyond

AWS Key Management Service (KMS) is a managed service that makes it easy for you to create and manage keys and control the use of encryption across a wide range of AWS services. KMS is a secure and resilient service that uses FIPS 140-2 validated hardware security modules to isolate and protect your keys.

You can create a key by clicking the button below.

Figure 2: AWS KMS Dashboard

We were then introduced to a wide variety of options such as the key type (symmetric or asymmetric), key usage, and other advanced options.


Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
voclabs/user1802981=Moore,_Tristan_@ 2026-0756-7185
KMS > Customer managed keys > Create key
Help me choose
Help me choose
Advanced options
Cancel
Next

Step 1 Configure key

Step 2 Add labels

Step 3 Define key administrative permissions

Step 4 Define key usage permissions

Step 5 Review

Configure key

Key type

● Symmetric
A single key used for encrypting and decrypting data or generating and verifying HMAC codes

○ Asymmetric
A public and private key pair used for encrypting and decrypting data or signing and verifying messages

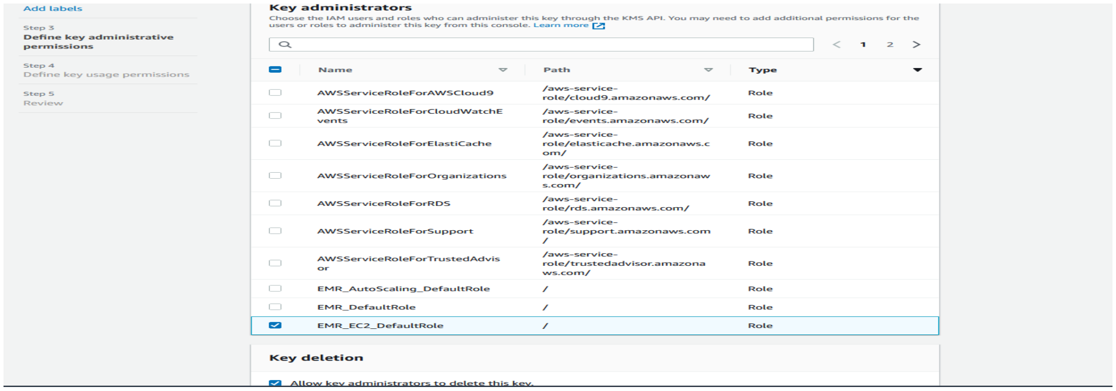
Key usage

● Encrypt and decrypt
Use the key only to encrypt and decrypt data.

○ Generate and verify MAC
Use the key only to generate and verify hash-based message authentication codes (HMAC).

Figure 3: AWS KMS Key Type and Key Usage

We were then able to allow specific AWS Identity and Access Management (IAM) groups and roles for users who were allowed have administrative permissions over this encryption key. Reference the figure below.



Add labels
Step 3
Define key administrative permissions
Step 4
Define key usage permissions
Step 5
Review

Key administrators

Choose the IAM users and roles who can administer this key through the KMS API. You may need to add additional permissions for the users or roles to administer this key from this console. Learn more

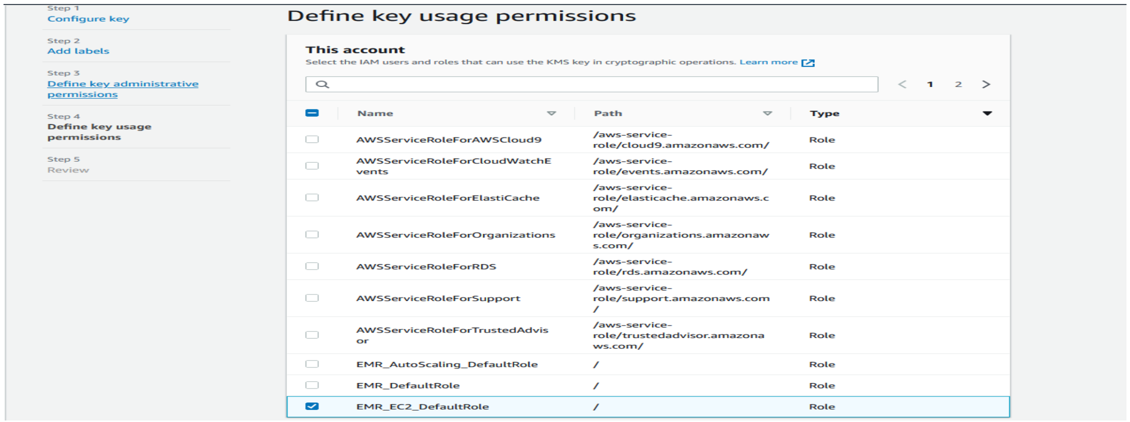
	Name	Path	Type
☐	AWSServiceRoleForAWSCloud9	/aws-service-role/cloud9.amazonaws.com/	Role
☐	AWSServiceRoleForCloudWatchEvents	/aws-service-role/events.amazonaws.com/	Role
☐	AWSServiceRoleForElastiCache	/aws-service-role/elasticache.amazonaws.com/	Role
☐	AWSServiceRoleForOrganizations	/aws-service-role/organizations.amazonaws.com/	Role
☐	AWSServiceRoleForRDS	/aws-service-role/rds.amazonaws.com/	Role
☐	AWSServiceRoleForSupport	/aws-service-role/support.amazonaws.com/	Role
☐	AWSServiceRoleForTrustedAdvisor	/aws-service-role/trustedadvisor.amazonaws.com/	Role
☐	EMR_AutoScaling_DefaultRole	/	Role
☐	EMR_DefaultRole	/	Role
☑	EMR_EC2_DefaultRole	/	Role

Key deletion

☑ Allow key administrators to delete this key.

Figure 4: KMS Key Admin Permissions

The next step in our configuration allowed us to assign AWS IAM groups and roles who were allowed to use the encryption key, but not have administrative privileges of the key.



Step 1
Configure key
Step 2
Add labels
Step 3
Define key administrative permissions
Step 4
Define key usage permissions
Step 5
Review

Define key usage permissions

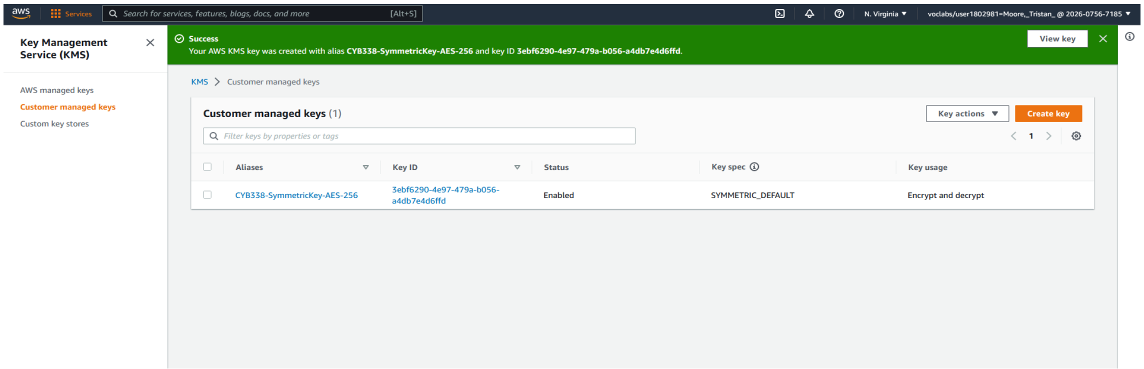
This account

Select the IAM users and roles that can use the KMS key in cryptographic operations. Learn more

	Name	Path	Type
☐	AWSServiceRoleForAWSCloud9	/aws-service-role/cloud9.amazonaws.com/	Role
☐	AWSServiceRoleForCloudWatchEvents	/aws-service-role/events.amazonaws.com/	Role
☐	AWSServiceRoleForElastiCache	/aws-service-role/elasticache.amazonaws.com/	Role
☐	AWSServiceRoleForOrganizations	/aws-service-role/organizations.amazonaws.com/	Role
☐	AWSServiceRoleForRDS	/aws-service-role/rds.amazonaws.com/	Role
☐	AWSServiceRoleForSupport	/aws-service-role/support.amazonaws.com/	Role
☐	AWSServiceRoleForTrustedAdvisor	/aws-service-role/trustedadvisor.amazonaws.com/	Role
☐	EMR_AutoScaling_DefaultRole	/	Role
☐	EMR_DefaultRole	/	Role
☑	EMR_EC2_DefaultRole	/	Role

Figure 5: KMS Key Usage Permissions

After this step, our key had been generated and is now visible within the AWS KMS dashboard.



aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
voclabs/user1802981=Moore,_Tristan_ @ 2026-0756-7185
Key Management Service (KMS)
AWS managed keys
Customer managed keys
Custom key stores
View key
KMS > Customer managed keys
Key actions
Create key
Filter keys by properties or tags

Success

Your AWS KMS key was created with alias **CYB338-SymmetricKey-AES-256** and key ID **3ebf6290-4e97-479a-b056-a4db7e4d6ffd**.

Customer managed keys (1)

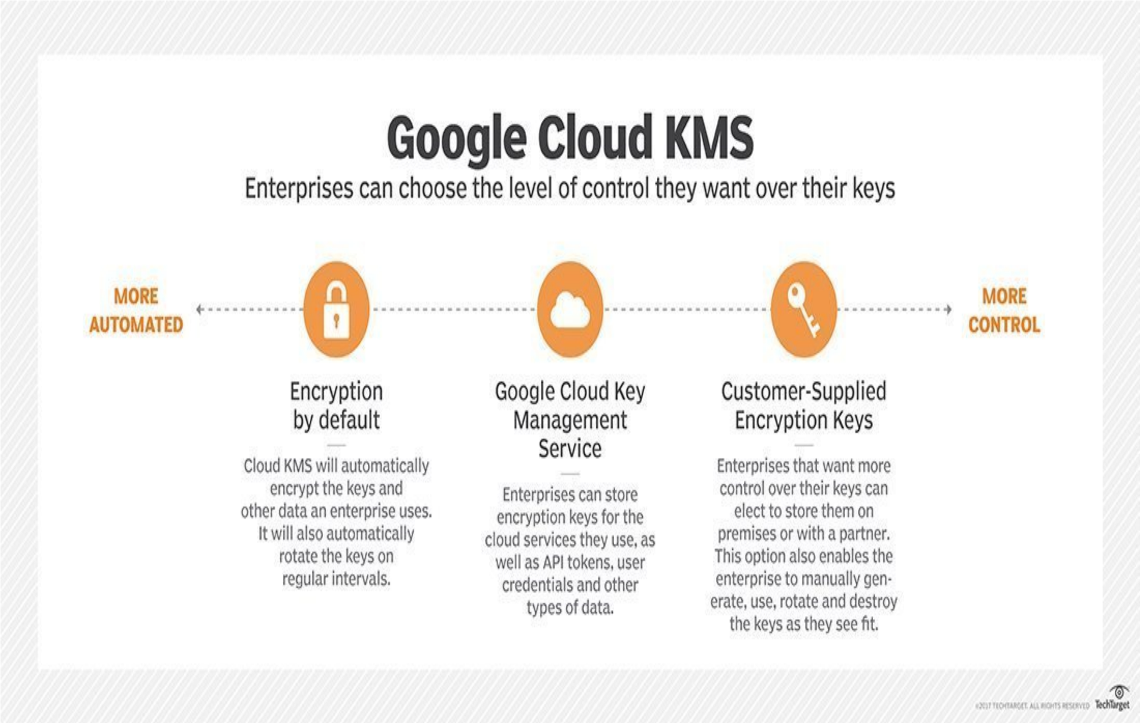
	Aliases	Key ID	Status	Key spec	Key usage
☐	CYB338-SymmetricKey-AES-256	3ebf6290-4e97-479a-b056-a4db7e4d6ffd	Enabled	SYMMETRIC_DEFAULT	Encrypt and decrypt

Figure 6: AWS KMS Dashboard after the KMS Key had been created

3.2 GOOGLE CLOUD PLATFORM (KMS)

Google Cloud Platform is a cloud hosted key management service that lets you manage symmetric and asymmetric cryptographic keys for cloud services. GCP can handle a variety of keys, these include AES 256, RSA 2048, RSA 3072, RSA 4096, EC P256, and EC P384 [41][42]. These can be protected via software or hardware based on user preference, and users can switch back and forth with a simple button click. Encryption keys can be managed by a third party as well using EKM (external key manager). These are deployed outside of Googles infrastructure and allows separation of data at rest and encryption keys. Using this EKM, users can request an encryption key, provide justification for said key, and a mechanism to either deny or approve the request [43].

Google KMS uses a five-level hierarchy. The top level is called GCP project which can be linked to an organization or company. After this comes keyrings, which hosts separate crypto keys. A key ring belongs to a certain project and therefore resides in a certain location. They also set permissions for the various keys they hold, so the keys within each key ring has the same permissions. These keys are subject to changes as the encryption changes. This is where the final tier comes in, CryptoKeyVersion. Google also offers a REST API as part of the KMS. This allows developers to access KMS functions to list, create, destroy, and update various encryption keys [44][45].



Google Cloud KMS

Enterprises can choose the level of control they want over their keys

Encryption by default

Cloud KMS will automatically encrypt the keys and other data an enterprise uses. It will also automatically rotate the keys on regular intervals.

Google Cloud Key Management Service

Enterprises can store encryption keys for the cloud services they use, as well as API tokens, user credentials and other types of data.

Customer-Supplied Encryption Keys

Enterprises that want more control over their keys can elect to store them on premises or with a partner. This option also enables the enterprise to manually generate, use, rotate and destroy the keys as they see fit.

Figure 7: GCP KMS summed up [44]

3.2.1 Google Cloud Platform Encryption

Google uses a very similar style of encryption for all their storage systems. While it follows the same style, the way it is implemented and rolled out varies with each system [46]. By default, GCP uses AES-256 encryption when data is at rest in storage. Data in transit is encrypted using TLS. By default, GCP uses Data Encryption Key (DEK) and Key-Encryption-Key (KEK). These two are paired and stored using Googles own Key Management Service (KMS). The KEK is used to encrypt the DEK, which was used to encrypt the actual data, to help increase security [47][48][49][50]. From here, the Google KMS is goes to work, using other services provided by Google to store the keys that are going to be used for decryption and further encryption on the cloud. To retrieve these keys, users must submit credentials/permissions proving they have rightful access. This is done using Identity and Access Management (IAM) [51].

GCP also provides another method to help further encryption. This is called Data Loss Prevention (DLP). DLP helps users identify possibly sensitive data and mask that data. Such data could include Personally Identifiable Information. By using the DLP method and combining it with the KMS, it is possible to use various encryption methods such as Format Preserving Encryption [52]. This means that the data is encrypted into an impossible to understand mess while the format is kept to the original plaintext. This method and many more can be accomplished using KMS, IAM, and DLP to help users further their encryption capabilities with their data. These services can also be setup to encrypt data automatically when uploaded to the Google cloud storage. Figure 8 shows how data is encrypted at Google. It begins by uploading the data, from there the data is chunked and encrypted separately. These encrypted chunks are then spread across Googles storage infrastructure.

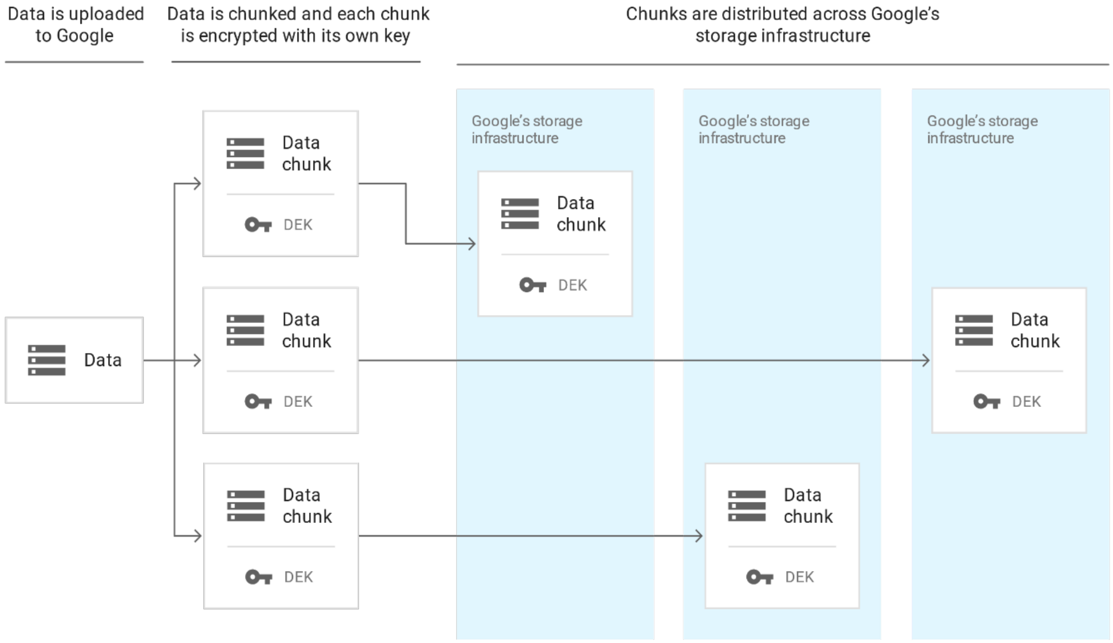
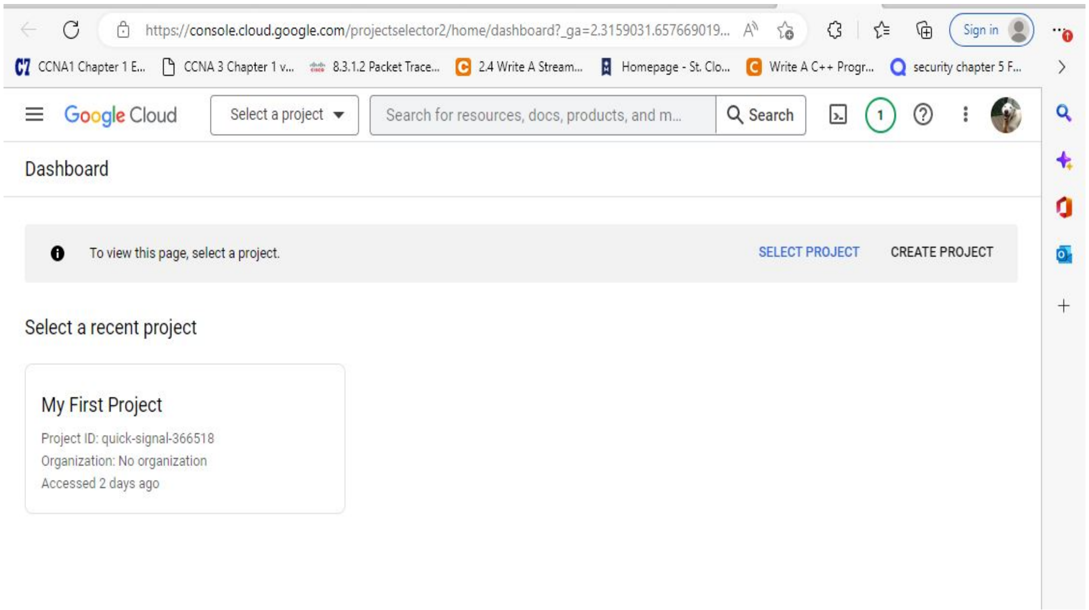

Data is uploaded to Google
Data is chunked and each chunk is encrypted with its own key
Chunks are distributed across Google's storage infrastructure
Google's storage infrastructure
Google's storage infrastructure
Google's storage infrastructure
Data
Data chunk
DEK
Data chunk
DEK
Data chunk
DEK
Data chunk
DEK
Data chunk
DEK
Data chunk
DEK

Figure 8: How data is encrypted on GCP [52]

3.2.2 Google Cloud Platform KMS Walk-Through

We are now going to show a walk-through of how to create a cryptographic key using GCP. The first step is to select the project from which the keyring will be held. We will be following the instructions found in [53].

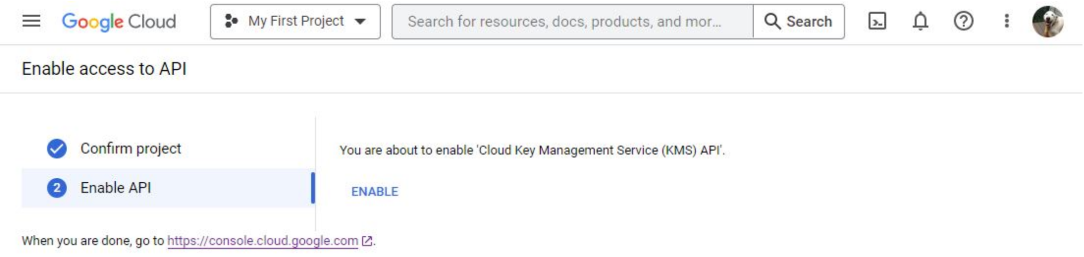


To view this page, select a project.

My First Project
Project ID: quick-signal-366518
Organization: No organization
Accessed 2 days ago

Figure 9: Selecting project to hold keys/keyring [53]

After the project has been selected, we need to make sure that some form of billing is enabled on the account to store the keys. We were using a 3-month trial so we were able to skip this step. After this we need to make sure the GCP KMS API is enabled.

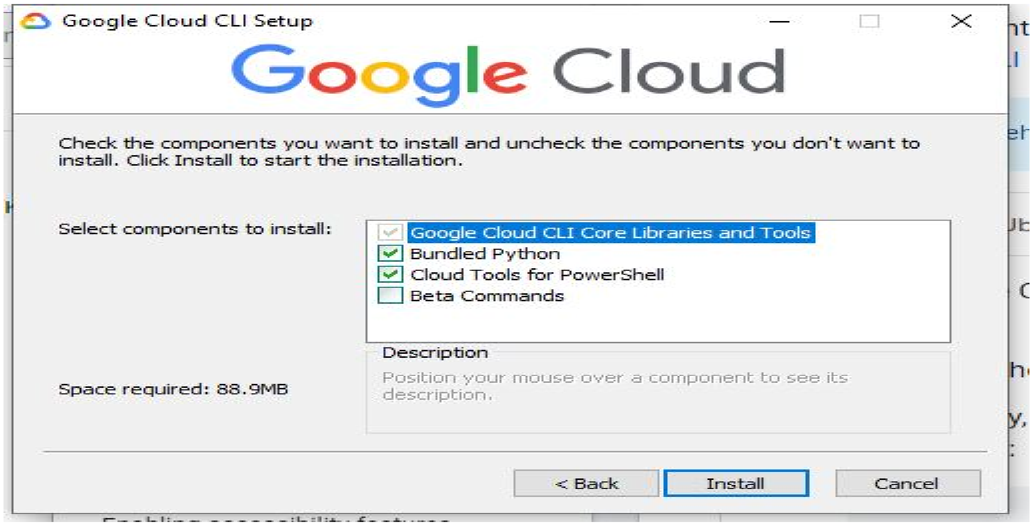


You are about to enable 'Cloud Key Management Service (KMS) API'.

When you are done, go to https://console.cloud.google.com.

Figure 10: Enabling the KMS API.

After enabling the API, we need to install and initialize GCP command line interface. This can be done through command line or a setup wizard. We used the setup wizard as it initializes the command line with the base configuration. Another reason why we used the setup wizard is because it also installs other dependencies that Google command line requires [54].



Check the components you want to install and uncheck the components you don't want to install. Click Install to start the installation.

Select components to install:

- ☑ Google Cloud CLI Core Libraries and Tools
- ☑ Bundled Python
- ☑ Cloud Tools for PowerShell
- □ Beta Commands

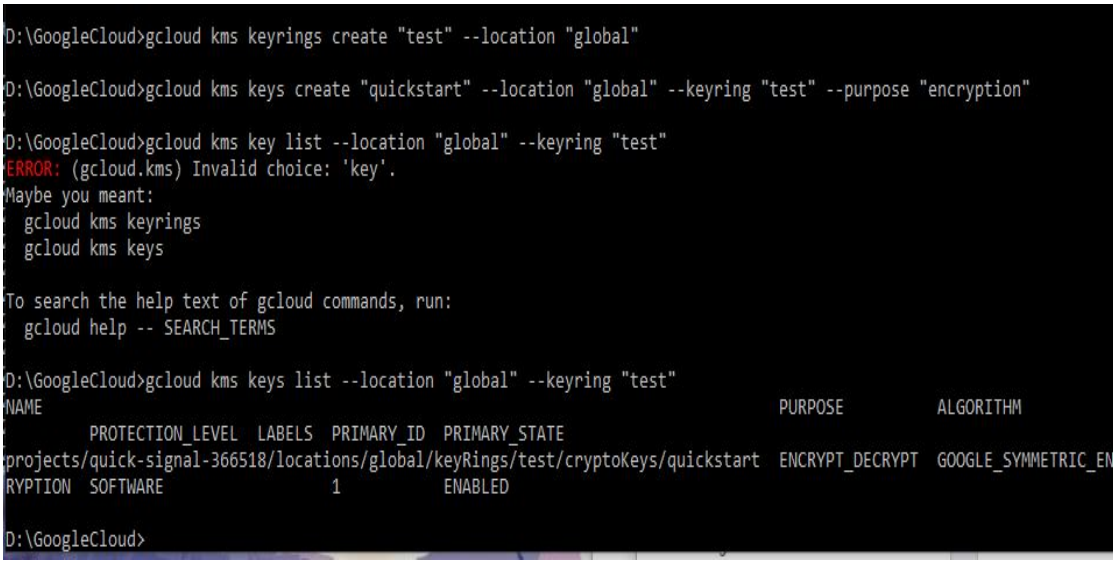
Description

Position your mouse over a component to see its description.

Space required: 88.9MB

Figure 11: Using setup wizard to install GCP CLI.

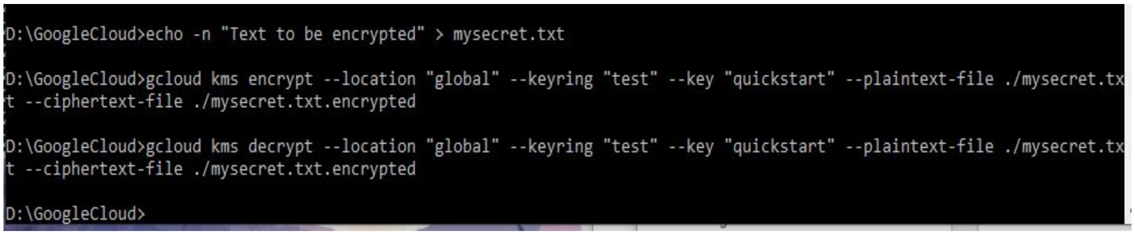
The next step is to create a keyring and key to encrypt data. We will make a keyring named test and a key named QuickStart. These are done using the following commands. The last gcloud command shows the metadata for the key just created. As shown in the path name, a keyring named test was created with a crypto key named QuickStart.



```
D:\GoogleCloud>gcloud kms keyrings create "test" --location "global"

D:\GoogleCloud>gcloud kms keys create "quickstart" --location "global" --keyring "test" --purpose "encryption"

D:\GoogleCloud>gcloud kms key list --location "global" --keyring "test"
ERROR: (gcloud.kms) Invalid choice: 'key'.
Maybe you meant:
  gcloud kms keyrings
  gcloud kms keys

To search the help text of gcloud commands, run:
  gcloud help -- SEARCH_TERMS

D:\GoogleCloud>gcloud kms keys list --location "global" --keyring "test"
NAME                                                                     PURPOSE          ALGORITHM
        PROTECTION_LEVEL  LABELS  PRIMARY_ID  PRIMARY_STATE
projects/quick-signal-366518/locations/global/keyRings/test/cryptoKeys/quickstart  ENCRYPT_DECRYPT  GOOGLE_SYMMETRIC_EN
RYPTION  SOFTWARE                 1           ENABLED

D:\GoogleCloud>
```

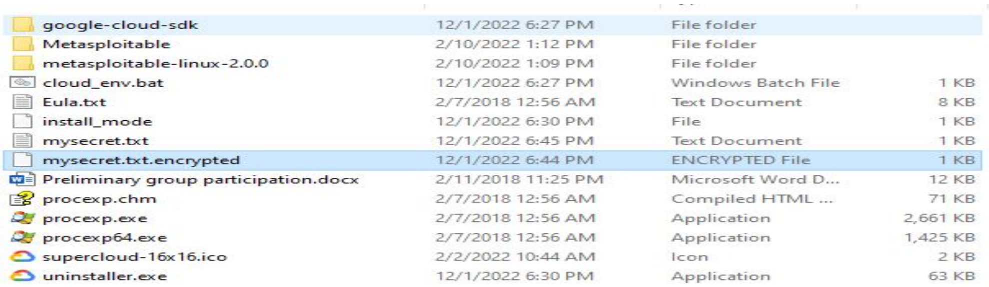
Figure 12: Creating a key ring and key named test and QuickStart respectively

Now we have created the keys, it is time to encrypt something. We used the echo command to send “Text to be encrypted” to a file named mysecret.txt. We then used the gcloud kms encrypt command, specifying the location, keyring, key, and plaintext/ciphertext file to input the unencrypted data from and output the encrypted data to. We finally encrypted something using the keys we created. Now it is time to decrypt the encrypted text. This is done through almost the same as the encrypt command, except we replace encrypt with decrypt. The rest of the parameters remain the same as the original encrypt command except for the cipher and plaintext locations are swapped.



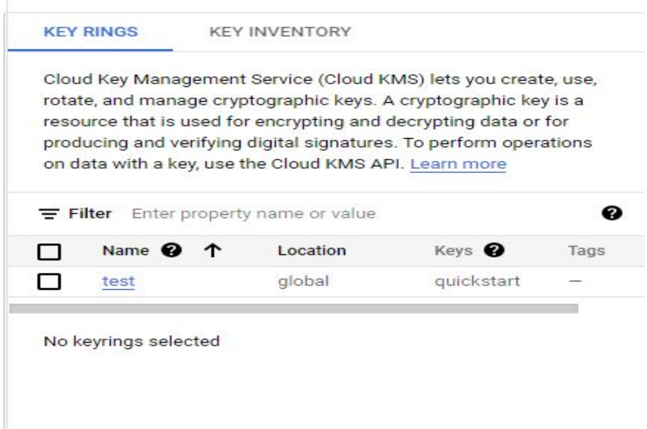
```
D:\GoogleCloud>echo -n "Text to be encrypted" > mysecret.txt

D:\GoogleCloud>gcloud kms encrypt --location "global" --keyring "test" --key "quickstart" --plaintext-file ./mysecret.tx
t --ciphertext-file ./mysecret.txt.encrypted

D:\GoogleCloud>gcloud kms decrypt --location "global" --keyring "test" --key "quickstart" --plaintext-file ./mysecret.tx
t --ciphertext-file ./mysecret.txt.encrypted

D:\GoogleCloud>
```

Figure 13: Creating text to be encrypted and encrypting it as well as decrypting



google-cloud-sdk	12/1/2022 6:27 PM	File folder	
Metasploitable	2/10/2022 1:12 PM	File folder	
metasploitable-linux-2.0.0	2/10/2022 1:09 PM	File folder	
cloud_env.bat	12/1/2022 6:27 PM	Windows Batch File	1 KB
Eula.txt	2/7/2018 12:56 AM	Text Document	8 KB
install_mode	12/1/2022 6:30 PM	File	1 KB
mysecret.txt	12/1/2022 6:45 PM	Text Document	1 KB
mysecret.txt.encrypted	12/1/2022 6:44 PM	ENCRYPTED File	1 KB
Preliminary group participation.docx	2/11/2018 11:25 PM	Microsoft Word D...	12 KB
procexp.chm	2/7/2018 12:56 AM	Compiled HTML ...	71 KB
procexp.exe	2/7/2018 12:56 AM	Application	2,661 KB
procexp64.exe	2/7/2018 12:56 AM	Application	1,425 KB
supercloud-16x16.ico	2/2/2022 10:44 AM	Icon	2 KB
uninstaller.exe	12/1/2022 6:30 PM	Application	63 KB

Figure 14: Encrypted file created



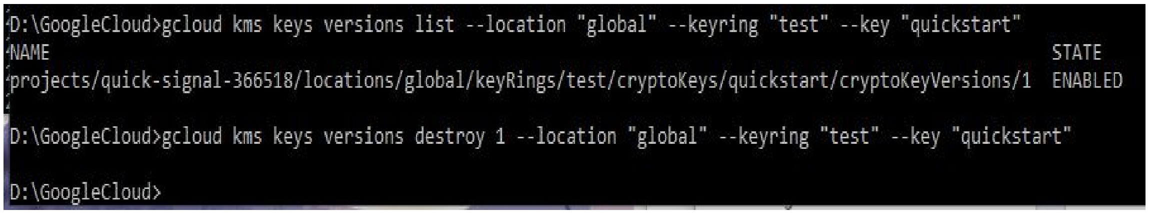
Cloud Key Management Service (Cloud KMS) lets you create, use, rotate, and manage cryptographic keys. A cryptographic key is a resource that is used for encrypting and decrypting data or for producing and verifying digital signatures. To perform operations on data with a key, use the Cloud KMS API. Learn more

□	Name	Location	Keys	Tags
□	test	global	quickstart	—

No keyrings selected

Figure 15: Keyring and key shown on GCP dashboard

Now we have successfully decrypted the file, we need to think about our keyring and the matter of storing it. We don't have much use for these keys and more, so we don't want to get charged for storage fees. To remedy this, we need to delete or destroy the keys. To do that we need the key version. This can be found using the KMS keys versions command. Our version was 1 so we input that into the KMS keys versions destroy command and specify which key we want to target.



```
D:\GoogleCloud>gcloud kms keys versions list --location "global" --keyring "test" --key "quickstart"
NAME                                                                                          STATE
projects/quick-signal-366518/locations/global/keyRings/test/cryptoKeys/quickstart/cryptoKeyVersions/1  ENABLED

D:\GoogleCloud>gcloud kms keys versions destroy 1 --location "global" --keyring "test" --key "quickstart"

D:\GoogleCloud>
```

Figure 16: Destroying the key we previously made

4 RESULTS

We were able to successfully create encryption keys in both AWS KMS and GCP KMS. Amazon Web Services offers a simple, yet granular Key Management Service that allowed us to leverage and manage encryption keys easily within the cloud-based environment. The Google Cloud Platform was relatively easy to use to create keys, and the command line that it offers is used for all Google Cloud resources. So it is very versatile as well as straight forward. Everything provided on the Google Cloud Console is also able to be altered through this command line, though it may be easier in most instances to just use the console. Both services offer a way to create an asymmetric or symmetric key.

In addition, this research shows how similar both systems are in how they work. Both utilize different method to encrypt the data key and use some of the same services to accomplish a task, such as Identity and Access Management (IAM). On the other hand, one of the main differences noticed between the two is that AWS handles key creation entirely through a GUI on the cloud dashboard. Meanwhile, GCP uses a specific command line interface to create keys and encrypt data. These keys can also be created through the console, though it is not as straightforward as it was with AWS. Other than how things are implemented; the actual process is relatively the same except for the concept of keyrings, which is unique to GCP it seems. One reason why this could be is because AWS asks while a key is being created for its permissions and who has access to it. GCP handles this by creating different keyrings to hold keys having different permissions with different accesses. Overall AWS seems to handle key creation in an easier, more graceful, way. This is mostly due to the fact a GUI is used instead of having to enter commands as the fastest method.

When it comes down to volume sizes available, AWS offers 500GB to 16 TB while GCP offers 1 GB to 64 TB. AWS offers different types of keys, those being regular data keys and customer master keys (CMK's). These master keys can be used to encrypt and decrypt data and the data keys are generated, encrypted, and decrypted by the CMK's. These can be customer or AWS managed. They essentially serve the same purpose as GCP's Key-Encryption-Key. GCP and AWS offer different encryption types. AWS offers AES-GCM and RSA-OAEP; while GCP offers RSA PKCS#1v1.5 and RSA-OAEP. Both also offer the same asymmetric key lengths. These are 2048-bit, 3072-bit, and 4096-bit RSA. It is the same for symmetric key length, that being 256-bit AES.

Another noticeable difference involves the way each of the two CSPs operate. AWS KMS is used to encrypt storage, services, and other resources within the cloud, while GCP KMS is utilized to encrypt data elsewhere and manage the keys within the cloud. This is largely due to the fact that GCP by default encrypts all data at-rest for their databases, data warehouses, storage buckets, and other storage services to take the burden off the customer to do so themselves.

5 CONCLUSION

Throughout this paper we have compared two different yet uniquely similar KMS and encryption systems. Both these systems seem to follow a similar guideline or path for how they encrypt their data and how they manage their keys. Even though they are very similar they still have their own unique qualities. Both services do a phenomenal job in encrypting data and managing their keys. It really is a user's preference as to what service they choose, as both offer almost the same service with a few variations. While it is unfortunate that we were unable to test Microsoft Azure KMS, it is not farfetched to believe that it also operates along similar principals to AWS and GCP. Both of these services can confidently say that they protect both the integrity of that data users provide to it and the keys created within.

References

[1] Mukherjee, Subhodeep, Venkataiah Chittipaka, Manish Mohan Baral, and Sharad Chandra Srivastava. "Integrating the challenges of cloud computing in supply chain management." In Recent Advances in Industrial Production: Select Proceedings of ICEM 2020, pp. 355-363. Springer Singapore, 2022.

[2] Singh, Nicholas, Kevin Bui, and Akalanka Mailewa. "Robust Efficiency Evaluation of NextCloud and GoogleCloud." Advances in Technology (2021): 536-545. (DOI:10.31357/ait.v1i2.5392)

[3] Wulf, Frederik, Tobias Lindner, Susanne Strahringer, and Markus Westner. "IaaS, PaaS, or SaaS? The Why of Cloud Computing Delivery Model Selection: Vignettes on the Post-Adoption of Cloud Computing." In Proceedings of the 54th Hawaii International Conference on System Sciences, 2021, pp. 6285-6294. 2021.

[4] Olaosebikan, Ayodeji, Thivanka PBM Dissanayaka, and Akalanka B. Mailewa. "Security & Privacy Comparison of NextCloud vs Dropbox: A Survey." In Midwest Instruction and Computing Symposium (MICS). 2022.

[5] Mailewa Dissanayaka, Akalanka, Roshan Ramprasad Shetty, Samip Kothari, Susan Mengel, Lisa Gittner, and Ravi Vadapalli. "A review of MongoDB and singularity container security in regards to hipaa regulations." In Companion Proceedings of the10th International Conference on Utility and Cloud Computing, pp. 91-97. 2017.

[6] Njuki, S., JIANBIAO ZHANG, EDNA TOO, and HAROLD BUKO DADYE. "Enhancing user data and VM security using the efficient hybrid of encrypting techniques." Journal of Theoretical and Applied Information Technology 97, no. 15 (2019).

[7] Ayuninggati, Tsara, Eka Purnama Harahap, and Raihan Junior. "Supply Chain Management, Certificate Management at the Transportation Layer Security in Charge of Security." Blockchain Frontier Technology 1, no. 01 (2021): 1-12.

[8] Ndri, Anna, Divya Bellamkonda, and Akalanka B. Mailewa. "Applications of Block-Chain Technologies to Enhance the Security of Intrusion Detection/Prevention Systems: A Review." In Midwest Instruction and Computing Symposium (MICS), vol. 2, p. 4. 2022.

[9] Kamal, Muhammad Ayoub, Hafiz Wahab Raza, Muhammad Mansoor Alam, and M. Mohd. "Highlight the features of AWS, GCP and Microsoft Azure that have an impact when choosing a cloud service provider." Int. J. Recent Technol. Eng 8, no. 5 (2020): 4124-4232.

[10] Deochake, Saurabh, and Vrushali Channapattan. "Identity and access management framework for multi-tenant resources in hybrid cloud computing." In Proceedings of the 17th International Conference on Availability, Reliability and Security, pp. 1-8. 2022.

[11] Dissanayaka, Akalanka Mailewa, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Security assurance of MongoDB in singularity LXCs: an elastic and convenient testbed using Linux containers to explore vulnerabilities." Cluster Computing 23 (2020): 1955-1971.

[12] Shamseddine, Maha, Wassim Itani, Auday Al-Dulaimy, and Javid Taheri. "Mitigating rogue node attacks in edge computing." In 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), pp. 1-6. IEEE, 2019.

[13] Ahmad, Shahnawaz, et al. “Cloud Security Framework and Key Management Services Collectively for Implementing DLP and IRM.” Materials Today : Proceedings, vol. 62, 2022, pp. 4828–36, https://doi.org/10.1016/j.matpr.2022.03.420.

[14] Khan, Saad, and Akalanka B. Mailewa. "Discover Botnets in IoT Sensor Networks: A Lightweight Deep Learning Framework with Hybrid Self-Organizing Maps." Microprocessors and Microsystems (2023): 104753. (DOI: https://doi.org/10.1016/j.micpro.2022.104753)

[15] Rozendaal, Kyle, and Akalanka Mailewa. "Neural Network Assisted IDS/IPS: An Overview of Implementations, Benefits, and Drawbacks." International Journal of Computer Applications 975: 8887. (DOI:10.5120/ijca2022922098)

[16] Muluve, Eva, Quentin Awori, Phanuel Owiti, Daniel Osuka, James Serembe, Paul Macharia, and Karen Austrian. "Using Mobile Biometrics and Management Information Systems to Enhance Quality and Accountability of Cash transfer in a Girls’ Empowerment Program in Rural and Urban Poor Settings." In 2020 IST-Africa Conference (IST-Africa), pp. 1-11. IEEE, 2020.

[17] Haghnegahdar, Lida, Sameehan S. Joshi, and Narendra B. Dahotre. "From IoT-based cloud manufacturing approach to intelligent additive manufacturing: Industrial Internet of Things—An overview." The International Journal of Advanced Manufacturing Technology (2022): 1-18.

[18] Dissanayaka, Akalanka Mailewa, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Vulnerability prioritization, root cause analysis, and mitigation of secure data analytic framework implemented with mongodb on singularity linux containers." In Proceedings of the 2020 the 4th International Conference on Compute and Data Analysis, pp. 58-66. 2020.

[19] Dissanayaka, Akalanka Mailewa, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Dynamic & portable vulnerability assessment testbed with Linux containers to ensure the security of MongoDB in Singularity LXCs." In Companion Conference of the Supercomputing-2018 (SC18). 2018.

[20] Noor, Talal, et al. “Trust Management of Services in Cloud Environments: Obstacles and Solutions.” ACM Computing Surveys, vol. 46, no. 1, 2013, pp. 1–30, https://doi.org/10.1145/2522968.2522980.

[21] Alshammari, Salah T., and Khalid Alsubhi. "Building a reputation attack detector for effective trust evaluation in a cloud services environment." Applied Sciences 11, no. 18 (2021): 8496.

[22] Gheisari, Mehdi, Hamid Esmaeili Najafabadi, Jafar A. Alzubi, Jiechao Gao, Guojun Wang, Aaqif Afzaal Abbasi, and Aniello Castiglione. "OBPP: An ontology-based framework for privacy-preserving in IoT-based smart city." Future Generation Computer Systems 123 (2021): 1-13.

[23] Khan, Muhammad Maaz Ali, Enow Nkongho Ehabe, and Akalanka B. Mailewa. "Discovering the Need for Information Assurance to Assure the End Users: Methodologies and Best Practices." In 2022 IEEE International Conference on Electro Information Technology (eIT), pp. 131-138. IEEE, May 2022. (DOI:10.1109/eIT53891.2022.9813791)

[24] Kaja, Durga Venkata Sowmya, Yasmin Fatima, and Akalanka B. Mailewa. "Data integrity attacks in cloud computing: A review of identifying and protecting techniques." Journal homepage: www. ijrpr. com ISSN 2582 (2022): 7421. (DOI:10.55248/gengpi.2022.3.2.8)

[25] Crișan-Mitra, Cătălina Silvia, Liana Stanca, and Dan-Cristian Dabija. "Corporate social performance: An assessment model on an emerging market." Sustainability 12, no. 10 (2020): 4077.

[26] Landi, Giovanni Catello, Francesca Iandolo, Antonio Renzi, and Andrea Rey. "Embedding sustainability in risk management: The impact of environmental, social, and governance ratings on corporate financial risk." Corporate Social Responsibility and Environmental Management 29, no. 4 (2022): 1096-1107.

[27] Kumar, Rakesh, and Rinkaj Goyal. "Performance based Risk driven Trust (PRTrust): On modeling of secured service sharing in peer-to-peer federated cloud." Computer Communications 183 (2022): 136-160.

[28] Jairu, Pankaj, and Akalanka B. Mailewa. "Network Anomaly Uncovering on CICIDS-2017 Dataset: A Supervised Artificial Intelligence Approach." In 2022 IEEE International Conference on Electro Information Technology (eIT), pp. 606-615. IEEE, May 2022. (DOI:10.1109/eIT53891.2022.9814045)

[29] Sapkota, Bhumika, and Akalanka B. Mailewa. "A Scalable Framework to Detect, Analyze, and Prevent Security Vulnerabilities in Enterprise Software-Defined Networks." Journal homepage: www. ijrpr. com ISSN 2582: 7421. (DOI:10.55248/gengpi.2022.3.2.1)

[30] Khayer, Abul, Md Shamim Talukder, Yukun Bao, and Md Nahin Hossain. "Cloud computing adoption and its impact on SMEs' performance for cloud supported operations: A dual-stage analytical approach." Technology in Society 60 (2020): 101225.

[31] Chahal, Rajanpreet Kaur, Neeraj Kumar, and Shalini Batra. "Trust management in social Internet of Things: A taxonomy, open issues, and challenges." Computer Communications 150 (2020): 13-46.

[32] Alemneh, Esubalew, Sidi-Mohammed Senouci, Philippe Brunet, and Tesfa Tegegne. "A two-way trust management system for fog computing." Future Generation Computer Systems 106 (2020): 206-220.

[33] Jorquera Valero, José María, Pedro Miguel Sánchez Sánchez, Manuel Gil Pérez, Alberto Huertas Celdrán, and Gregorio Martinez Perez. "Cutting-Edge Assets for Trust in 5G and Beyond: Requirements, State of the Art, Trends, and Challenges." ACM Computing Surveys 55, no. 11 (2023): 1-36.

[34] Gamnis, Steven, Matthew VanderLinden, and Akalanka Mailewa. "Analyzing Data Encryption Efficiencies for Secure Cloud Storages: A Case Study of Pcloud vs OneDrive vs Dropbox." Advances in Technology (2022): 79-98. (DOI:10.31357/ait.v2i1.5526)

[35] Garfinkel, Simson L., and Philip Leclerc. "Randomness concerns when deploying differential privacy." In Proceedings of the 19th Workshop on Privacy in the Electronic Society, pp. 73-86. 2020.

[36] Saarinen, Markku-Juhani O. "Mobile energy requirements of the upcoming NIST post-quantum cryptography standards." In 2020 8th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 23-30. IEEE, 2020.

[37] Mailewa, Akalanka, Susan Mengel, Lisa Gittner, and Hafiz Khan. "Mechanisms and techniques to enhance the security of big data analytic framework with mongodb and Linux containers." Array 15 (2022): 100236. (DOI:10.1016/j.array.2022.100236)

[38] Raso, Emanuele, Lorenzo Bracciale, Pierpaolo Loreti, and Giuseppe Bianchi. "ABEBox: A data driven access control for securing public cloud storage with efficient key revocation." In Proceedings of the 16th International Conference on Availability, Reliability and Security, pp. 1-7. 2021.

[39] Jarecki, Stanislaw, Hugo Krawczyk, and Jason Resch. "Updatable oblivious key management for storage systems." In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 379-393. 2019.

[40] Kamaraju, Ashvin, Asad Ali, and Rohini Deepak. "Best Practices for Cloud Data Protection and Key Management." In Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3, pp. 117-131. Springer International Publishing, 2022.

[41] Guptha, Ashwin, Harshaan Murali, and T. Subbulakshmi. "A Comparative Analysis of Security Services in Major Cloud Service Providers." In 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 129-136. IEEE, 2021.

[42] Mailewa, Akalanka, and Kyle Rozendaal. "A Novel Method for Moving Laterally and Discovering Malicious Lateral Movements in Windows Operating Systems: A Case Study." Advances in Technology (2022): 291-321, ISSN 2773-7098. (DOI:10.31357/ait.v2i3.5584)

[43] ACHAR, SANDESH, HRISHITVA PATEL, and SANWAL HUSSAIN. "DATA SECURITY IN CLOUD: A REVIEW." Asian Journal of Advances in Research (2022): 76-83.

[44] Roy, Agniswar, Abhik Banerjee, and Navneet Bhardwaj. "A Study on Google Cloud Platform (GCP) and Its Security." Machine Learning Techniques and Analytics for Cloud Security (2021): 313-338.

[45] Mailewa, Akalanka, and Jayantha Herath. "Operating Systems Learning Environment with VMware" In The Midwest Instruction and Computing Symposium. Retrieved from http://www.micsymposium.org/mics2014/ProceedingsMICS_2014/mics2014_submission_14.pdf. 2014.

[46] Dageville, Benoit, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh et al. "The snowflake elastic data warehouse." In Proceedings of the 2016 International Conference on Management of Data, pp. 215-226. 2016.

[47] Mbae, Oscar, David Mwathi, and Edna Too. "Secure Cloud Based Approach for Mobile Devices User Data." Open Access Library Journal 9, no. 9 (2022): 1-20.

[48] Shetty, Roshan Ramprasad, Akalanka Mailewa Dissanayaka, Susan Mengel, Lisa Gittner, Ravi Vadapalli, and Hafiz Khan. "Secure NoSQL based medical data processing and retrieval: the exposome project." In Companion Proceedings of the10th International Conference on Utility and Cloud Computing, pp. 99-105. 2017.

[49] Simkhada, Emerald, Elisha Shrestha, Sujan Pandit, Upasana Sherchand, and Akalanka Mailewa Dissanayaka. "Security threats/attacks via botnets and botnet detection & prevention techniques in computer networks: a review." In The Midwest Instruction and Computing Symposium.(MICS), North Dakota State University, Fargo, ND. 2019.

[50] Akintaro, Mojolaoluwa, Teddy Pare, and Akalanka Mailewa Dissanayaka. "Darknet and black market activities against the cybersecurity: a survey." In The Midwest Instruction and Computing Symposium.(MICS), North Dakota State University, Fargo, ND. 2019.

[51] Chandramouli, Ramaswamy, Michaela Iorga, and Santosh Chokhani. "Cryptographic key management issues and challenges in cloud services." Secure Cloud Computing (2013): 1-30.

[52] Ruiz Díaz, Blanca. "Deployment of a lab environment to identify and protect sensitive data in the cloud." Bachelor's thesis, Universitat Politècnica de Catalunya, 2022.

[53] Khalil, Maad M., Sergey E. Adadurov, and M. Sh Mahmood. "Mastering Google cloud: building the platform that serves your needs." Models and Methods for Researching Information Systems in Transport 2020 (MMRIST 2020) 1 (2020): 41-46.

[54] Carvalho, Daniel, João Morais, João Almeida, Pedro Martins, Carlos Quental, and Filipe Caldeira. "A Technical Overview on the Usage of Cloud Encryption Services." In European Conference on Cyber Warfare and Security, pp. 733-XI. Academic Conferences International Limited, 2019.