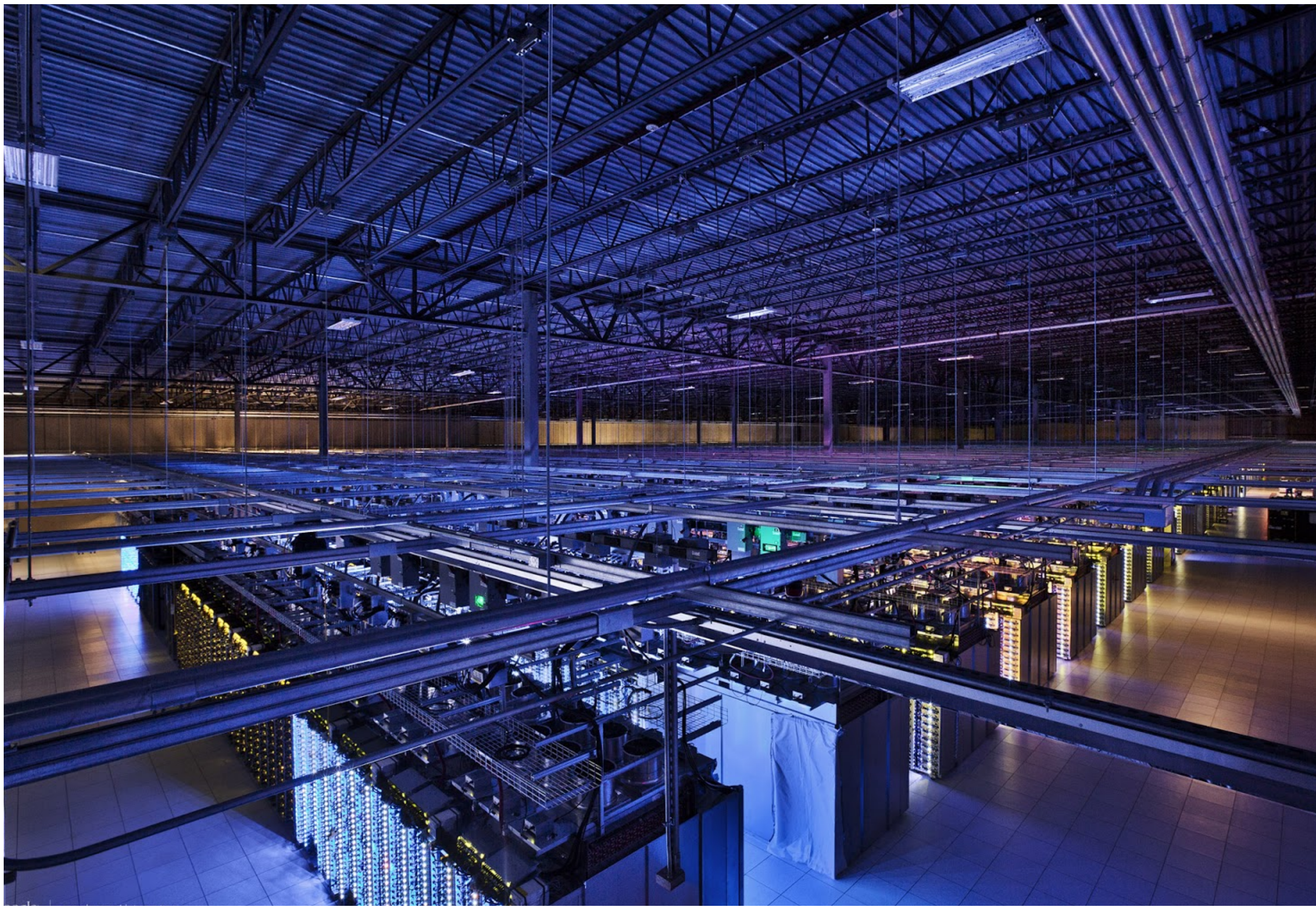
Google Cloud Whitepaper
July 2020

Encryption at rest in Google Cloud



Google Cloud

CIO-level summary

- Google uses several layers of encryption to protect customer data at rest in Google Cloud products.

- Google Cloud encrypts all customer content stored at rest, without any action required from the customer, using one or more encryption mechanisms.

- Data for storage is split into chunks, and each chunk is encrypted with a unique data encryption key. These data encryption keys are stored with the data, encrypted with ("wrapped" by) key encryption keys that are exclusively stored and used inside Google's central Key Management Service. Google's Key Management Service is redundant and globally distributed.

- All data stored in Google Cloud is encrypted at the storage level using AES256, with the exception of a small number of Persistent Disks created before 2015 that use AES128.

- Google uses a common cryptographic library, Tink, which incorporates our FIPS 140-2 validated module, BoringCrypto, to implement encryption consistently across almost all Google Cloud products. Consistent use of a common library means that only a small team of cryptographers needs to implement and maintain this tightly controlled and reviewed code.

Disclaimer

Introduction

For many individuals and companies, security is a deciding factor in choosing a public cloud vendor. At Google, security is of the utmost importance. We take security and privacy seriously, and we work tirelessly to protect your data—whether it is traveling over the internet, moving between our data centers, or stored on our servers.

Central to our comprehensive security strategy is encryption in transit and at rest, which ensures the data can be accessed only by the authorized roles and services with audited access to the encryption keys. This paper describes Google's approach to encryption at rest for the Google Cloud, and how Google uses it to keep your information more secure.

This document is targeted at CISOs and security operations teams currently using or considering using Google Cloud. With the exception of the introduction, this document assumes a basic understanding of encryption and cryptographic primitives.

What is encryption

Encryption is a process that takes legible data as input (often called plaintext), and transforms it into an output (often called ciphertext) that reveals little or no information about the plaintext. The encryption algorithm used is public, such as the Advanced Encryption Standard (AES), but execution depends on a key, which is kept secret. To decrypt the ciphertext back to its original form, you need to employ the key. At Google, the use of encryption to keep data confidential is usually combined with integrity protection; someone with access to the ciphertext can neither understand it nor make a modification without knowledge of the key. For more information on cryptography, a good resource is an Introduction to Modern Cryptography.

In this whitepaper, we focus on encryption at rest. By encryption at rest, we mean encryption used to protect data that is stored on a disk (including solid-state drives) or backup media. For information about encryption of data in transit, see our Encryption in Transit in Google Cloud whitepaper.

Why encryption helps secure customer data

Encryption is one piece of a broader security strategy. Encryption adds a layer of defense in depth for protecting data—encryption ensures that if the data accidentally falls into an attacker's hands, they cannot access the data without also having access to the encryption keys. Even if an attacker obtains the storage devices containing your data, they won't be able to understand or decrypt it.

Encryption at rest reduces the surface of attack by effectively "cutting out" the lower layers of the hardware and software stack. Even if these lower layers are compromised (for example, through physical access to devices), the data on those devices is not compromised if adequate encryption is deployed. Encryption also acts as a "chokepoint"—centrally managed encryption keys create a single place where access to data is enforced and can be audited.

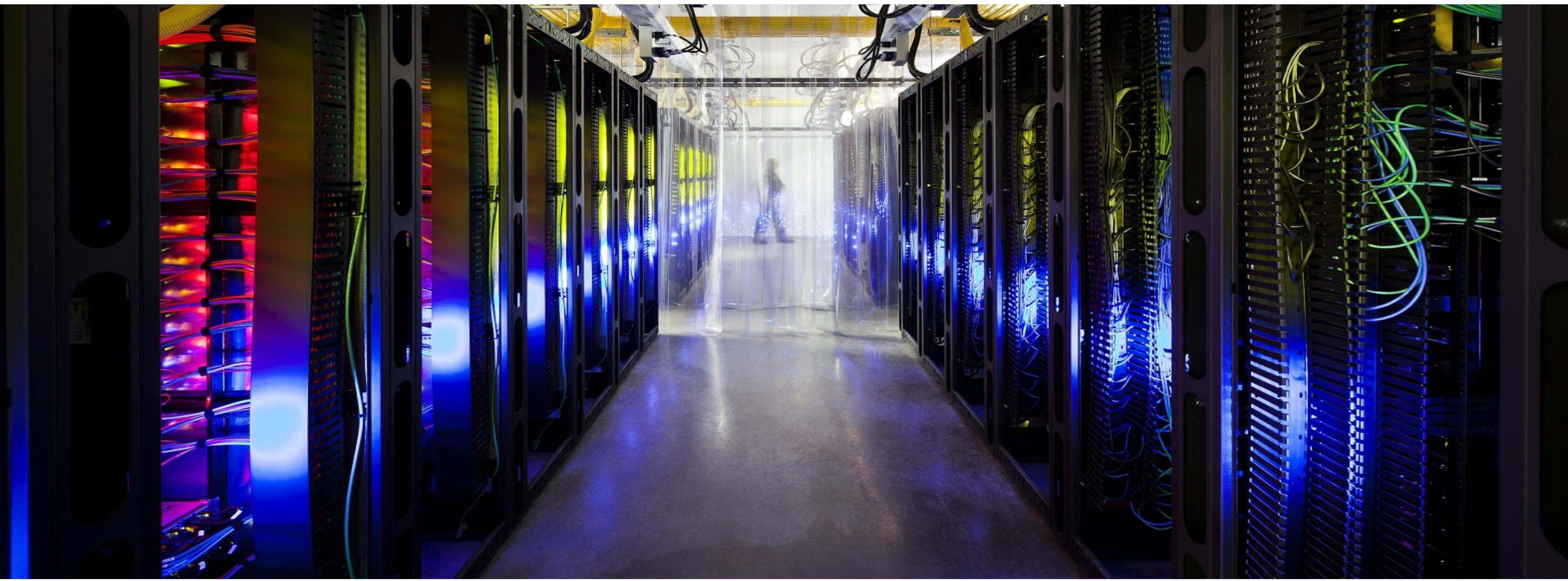
Encryption provides an important mechanism in how Google ensures the privacy of customer data—it allows systems to manipulate data, for example, for backup, and engineers to support our infrastructure, without providing access to content.

What we consider customer data

As defined in the Google Cloud terms of service, customer data refers to content provided to Google by a Google Cloud customer (or at their direction), directly or indirectly, via Google Cloud services used by that customer's account. Customer data includes customer content and customer metadata.

Customer content is data that Google Cloud customers generate themselves or provide to Google, like data stored in Cloud Storage, disk snapshots used by Compute Engine, and Identity and Access Management policies. The encryption at rest of customer content is the focus of this whitepaper.

Customer metadata makes up the rest of customer data and refers to all data that cannot be classified as customer content. This could include auto-generated project numbers, timestamps, and IP addresses, as well as the byte size of an object in Cloud Storage, or the machine type in Compute Engine. Metadata is protected to a degree that is reasonable for ongoing performance and operations.



Google's default encryption

Encryption of data at rest

Google Cloud encrypts all customer content stored at rest, without any action from the customer, using one or more encryption mechanisms.

Layers of encryption

Google uses several layers of encryption to protect data. Using multiple layers of encryption adds redundant data protection and allows us to select the optimal approach based on application requirements.

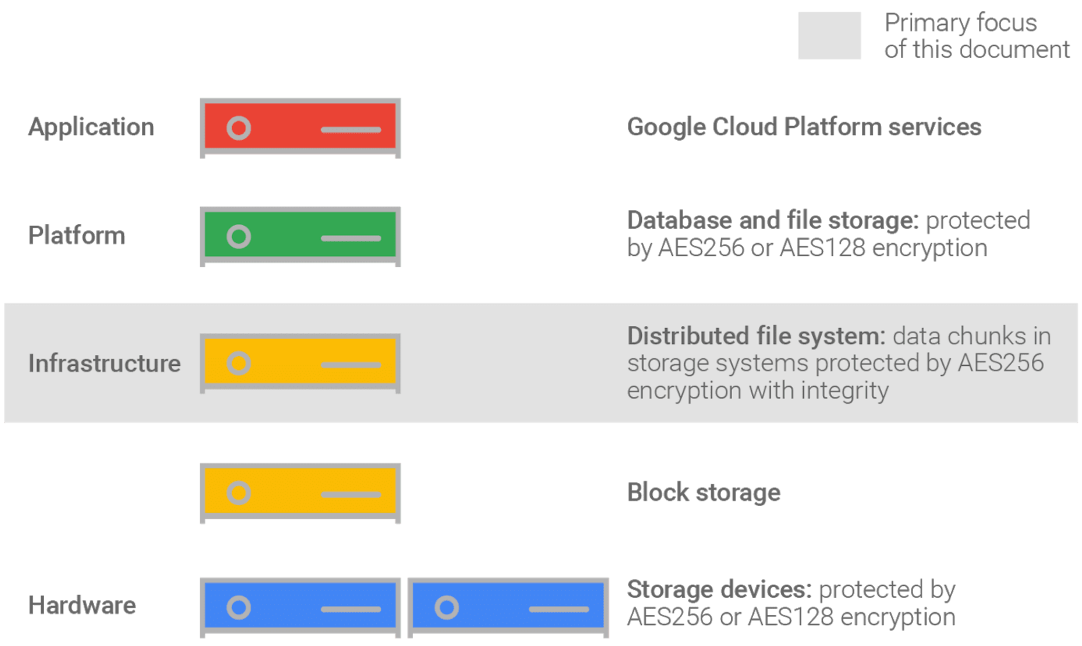

Primary focus of this document
Application
Google Cloud Platform services
Platform
Database and file storage: protected by AES256 or AES128 encryption
Infrastructure
Distributed file system: data chunks in storage systems protected by AES256 encryption with integrity
Block storage
Hardware
Storage devices: protected by AES256 or AES128 encryption

Figure 1: Several layers of encryption are used to protect data stored in Google Cloud. Either distributed file system encryption or database and file storage encryption is in place for almost all files; and storage device encryption is in place for almost all files.

Encryption at the storage system layer

To understand how specifically Cloud Storage encryption works, it's important to understand how Google stores customer data. Data is broken into subfile chunks for storage; each chunk can be up to several GB in size. Each chunk is encrypted at the storage level with an individual encryption key: two chunks will not have the same encryption key, even if they are part of the same Cloud Storage object, owned by the same customer, or stored on the same machine.[1] If a chunk of data is updated, it is encrypted with a new key, rather than by reusing the existing key. This partition of data, each using a different key, means the "blast radius" of a potential data encryption key compromise is limited to only that data chunk.

Google encrypts data prior to it being written to disk. Encryption is inherent in all of Google's storage systems—rather than added on afterward.

Each data chunk has a unique identifier. Access control lists (ACLs) ensure that each chunk can be decrypted only by Google services operating under authorized roles, which are granted access at that point in time. This prevents access to the data without authorization, bolstering both data security and privacy.

Each chunk is distributed across Google's storage systems and replicated in encrypted form for backup and disaster recovery. A malicious individual who wanted to access customer data would need to know and be able to access (1) all storage chunks corresponding to the data they want, and (2) the encryption keys corresponding to the chunks.

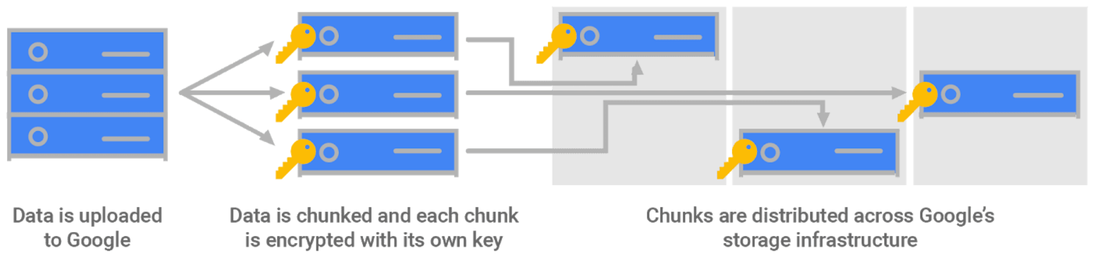

Data is uploaded to Google
Data is chunked and each chunk is encrypted with its own key
Chunks are distributed across Google's storage infrastructure

Figure 2: Data at Google is broken up into encrypted chunks for storage.

Google uses the Advanced Encryption Standard (AES) algorithm to encrypt data at rest. All data at the storage level is encrypted with AES256 by default, with the exception of a small number of Persistent Disks created prior to 2015 that use AES128. AES is widely used because (1) both AES256 and AES128 are recommended by the National Institute of Standards and Technology (NIST) for long-term storage use (as of March 2019), and (2) AES is often included as part of customer compliance requirements.

Data stored across Cloud Storage is encrypted at the storage level using AES, in Galois/Counter Mode (GCM) in almost all cases. In select cases, AES is used in Cipher Block Chaining (CBC) mode with a hashed message authentication code (HMAC) for authentication; and for some replicated files, AES is

[1] A data chunk in Datastore, App Engine, and Pub/Sub may contain data of multiple customers. See the section on granularity of data encryption by service.

used in Counter (CTR) mode with HMAC. (Further details on algorithms are provided later in this document.) In other Google Cloud products, AES is used in a variety of modes.

Encryption at the storage device layer

In addition to the storage system level encryption described above, in most cases data is also encrypted at the storage device level with AES256 for hard disks (HDD) and solid state drives (SSD), using a separate device-level key (which is different than the key used to encrypt the data at the storage level). A small number of legacy HDDs use AES128. SSDs used by Google Cloud implement AES256 for user data exclusively.

Encryption of backups

Google's backup system ensures that data remains encrypted throughout the backup process. This approach avoids unnecessarily exposing plaintext data.

In addition, the backup system further encrypts each backup file independently with its own data encryption key (DEK), derived from a key stored in Google's Key Management Service (KMS) plus a randomly generated per-file seed at backup time. Another DEK is used for all metadata in backups, which is also stored in Google's KMS. (Further information on key management is in a later section.)

FIPS compliance for data at rest

Google Cloud uses a FIPS 140-2 validated encryption module (certificate 3318) in our production environment.

Key management

Data encryption keys, key encryption keys, and Google's Key Management Service

The key used to encrypt the data in a chunk is called a data encryption key (DEK). Because of the high volume of keys at Google, and the need for low latency and high availability, these keys are stored near the data that they encrypt. The DEKs are encrypted with (or "wrapped" by) a key encryption key (KEK). One or more KEKs exist for each Google Cloud service. These KEKs are stored centrally in Google's KMS, a repository built specifically for storing keys. Having a smaller number of KEKs than DEKs and using a central key management service makes storing and encrypting data at Google scale manageable, and allows us to track and control data access from a central point.

For each Google Cloud customer, any non-shared resources[2] are split into data chunks and encrypted with keys separate from keys used for other customers.[3] These DEKs are even separate from those that protect other pieces of the same data owned by that same customer.

DEKs are generated by the storage system using Google's common cryptographic library. They are then sent to KMS to wrap with that storage system's KEK, and the wrapped DEKs are passed back to the storage system to be kept with the data chunks. When a storage system needs to retrieve encrypted data, it retrieves the wrapped DEK and passes it to KMS. KMS then verifies that this service is authorized to use the KEK, and if so, unwraps and returns the plaintext DEK to the service. The service then uses the DEK to decrypt the data chunk into plaintext and verify its integrity.

Most KEKs for encrypting data chunks are generated within KMS, and the rest are generated inside the storage services. For consistency, all KEKs are generated using Google's common cryptographic library, using a random number generator (RNG) built by Google. This RNG is based on NIST 800-90Ar1 CTR-DRBG and generates an AES256 KEK.[4] This RNG is seeded from the Linux kernel's RNG, which in turn is seeded from multiple independent entropy sources. This includes entropic events from the data center environment, such as fine-grained measurements of disk seeks and inter-packet arrival times, and Intel's RDRAND instruction where it is available (on Ivy Bridge and newer CPUs).

Data stored in Google Cloud is encrypted with DEKs using AES256 or AES128, as described above; and any new data encrypted in persistent disks in Compute Engine is encrypted using AES256. DEKs are wrapped with KEKs using AES256 or AES128, depending on the Google Cloud service. We are currently working on upgrading all KEKs for Cloud services to AES256.

[2] An example of a shared resource (where this segregation does not apply) would be a shared base image in Compute Engine—naturally, multiple customers refer to a single copy, which is encrypted by a single DEK.
[3] With the exception of data stored in Datastore, App Engine, and Pub/Sub, where more than one customer's data may be encrypted with the same DEK. See the section on granularity of data encryption by service.
[4] Note that in the past, this was AES128, and some of these keys remain active for decrypting data.

Google's KMS manages KEKs and was built solely for this purpose. It was designed with security in mind. KEKs are not exportable from Google's KMS by design; all encryption and decryption with these keys must be done within KMS. This helps prevent leaks and misuse, and enables KMS to emit an audit trail when keys are used.

KMS can automatically rotate KEKs at regular time intervals, using Google's common cryptographic library to generate new keys. Though we often refer to just a single key, we really mean that data is protected using a key set: one key active for encryption and a set of historical keys for decryption, the number of which is determined by the key rotation schedule. The actual rotation schedule for a KEK varies by service, but the standard rotation period is 90 days. Cloud Storage specifically rotates its KEKs every 90 days, and can store up to 20 versions, requiring re-encryption of data at least once every five years (though in practice, data re-encryption is much more frequent). KMS-held keys are backed up for disaster recovery purposes, and they are indefinitely recoverable.

The use of KEKs is managed by access control lists (ACLs) in KMS for each key, with a per-key policy. Only authorized Google services and users are allowed access to a key. The use of each key is tracked at the level of the individual operation that requires that key—so every time an individual uses a key, it is authenticated and logged. All human data accesses are auditable as part of Google's overall security and privacy policies.

When a Google Cloud service accesses an encrypted chunk of data, here's what happens:

1. The service makes a call to the storage system for the data it needs.
2. The storage system identifies the chunks in which that data is stored (the chunk IDs) and where they are stored.
3. For each chunk, the storage system pulls the wrapped DEK stored with that chunk (in some cases, this is done by the service), and sends it to KMS for unwrapping.
4. The storage system verifies that the identified job is allowed to access that data chunk based on a job identifier, and using the chunk ID; and KMS verifies that the storage system is authorized both to use the KEK associated with the service, and to unwrap that specific DEK.
5. KMS does one of the following:
    - Passes the unwrapped DEK back to the storage system, which decrypts the data chunk and passes it to the service. Or, in some rare cases,
    - Passes the unwrapped DEK to the service; the storage system passes the encrypted data chunk to the service, which decrypts the data chunk and uses it.

This process is different in dedicated storage devices, such as local SSDs, where the device manages and protects the device-level DEK.

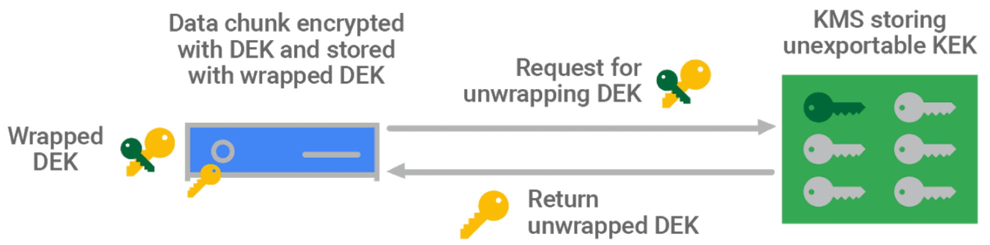

Data chunk encrypted with DEK and stored with wrapped DEK
KMS storing unexportable KEK
Request for unwrapping DEK
Wrapped DEK
KEK
Return unwrapped DEK

Figure 3: To decrypt a data chunk, the storage service calls Google's Key Management Service (KMS) to retrieve the unwrapped data encryption key (DEK) for that data chunk.

Encryption key hierarchy and root of trust

Google's KMS is protected by a root key called the KMS master key, which wraps all the KEKs in KMS. This KMS master key is AES256[5] and is itself stored in another key management service, called the Root KMS. Root KMS stores a much smaller number of keys—approximately a dozen. For additional security, Root KMS is not run on general production machines, but instead is run only on dedicated machines in each Google data center.

Root KMS in turn has its own root key, called the root KMS master key, which is also AES256[6] and is stored in a peer-to-peer infrastructure, the root KMS master key distributor, which replicates these keys globally. The root KMS master key distributor only holds the keys in RAM on the same dedicated machines as Root KMS, and uses logging to verify proper use. One instance of the root KMS master key distributor runs for every instance of Root KMS. (The root KMS master key distributor is still being phased in, to replace a system that operated in a similar manner but was not peer to peer.)

When a new instance of the root KMS master key distributor is started, it is configured with a list of host names already running distributor instances. Distributor instances can then obtain the root KMS master key from other running instances. Other than the disaster-recovery mechanisms described below, the root KMS master key exists only in RAM on a limited number of specially secured machines.

To address the scenario where all instances of the root KMS master key distributor restart simultaneously, the root KMS master key is also backed up on secure hardware devices stored in physical safes in highly secured areas in two physically separated, global Google locations. This backup would be needed only if all distributor instances were to go down at once; for example, in a global restart. Fewer than 20 Google employees are able to access these safes.

[5] Note that in the past, this was AES128, and some of these keys remain active for decrypting data.
[6] Note that in the past, this was AES128, and some of these keys remain active for decrypting data.

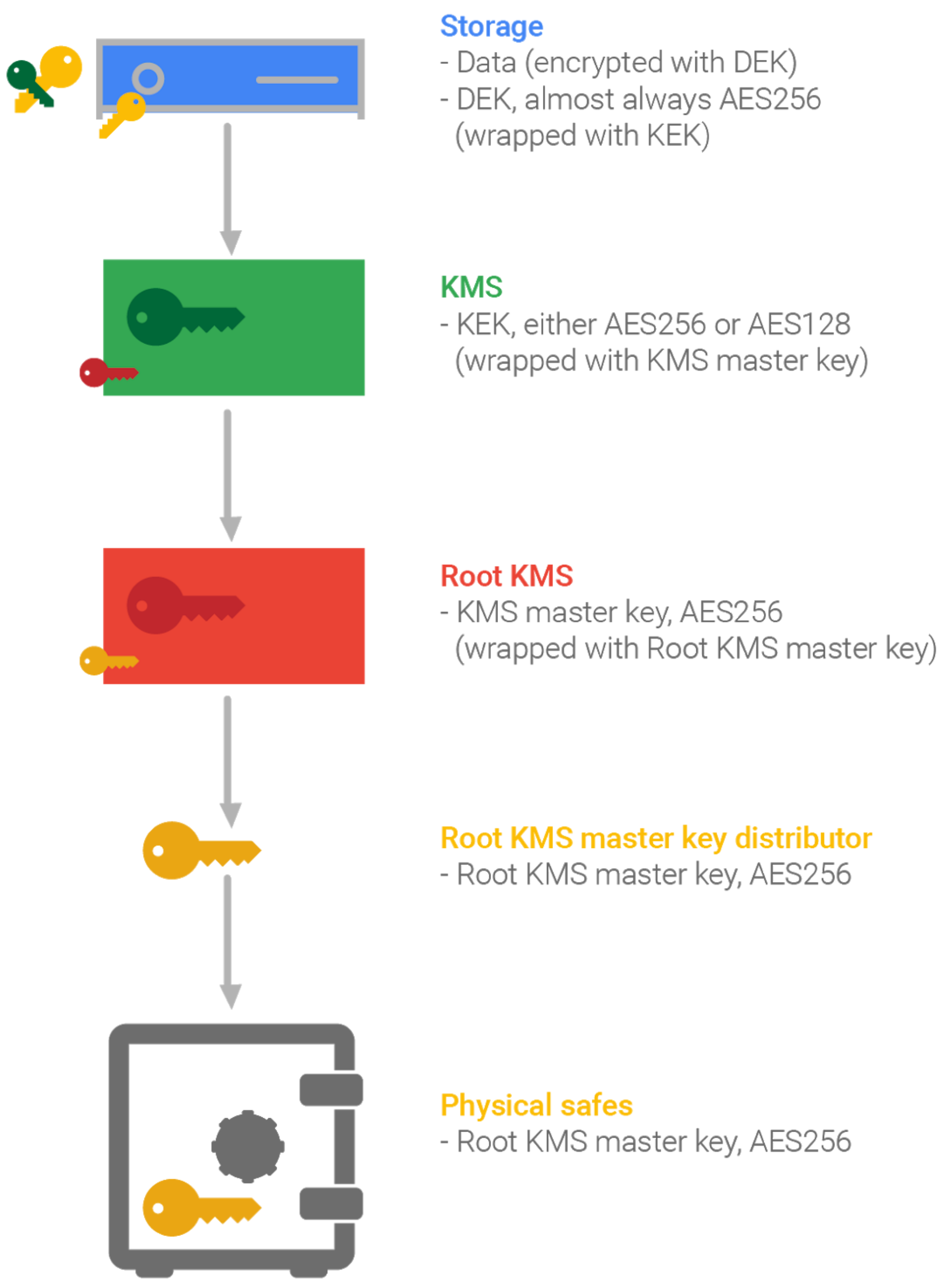

Storage
- Data (encrypted with DEK)
- DEK, almost always AES256 (wrapped with KEK)
KMS
- KEK, either AES256 or AES128 (wrapped with KMS master key)
Root KMS
- KMS master key, AES256 (wrapped with Root KMS master key)
Root KMS master key distributor
- Root KMS master key, AES256
Physical safes
- Root KMS master key, AES256

Figure 4: The encryption key hierarchy protects a chunk of data with a DEK, wrapped with a KEK in KMS, which is in turn protected by Root KMS and the root KMS master key distributor.

To summarize:

- Data is chunked and encrypted with DEKs.
- DEKs are encrypted with KEKs.
- KEKs are stored in KMS.
- KMS is run on multiple machines in data centers globally.
    - KMS keys are wrapped with the KMS master key, which is stored in Root KMS.
- Root KMS is much smaller than KMS and runs only on dedicated machines in each data center.
    - Root KMS keys are wrapped with the root KMS master key, which is stored in the root KMS master key distributor.
- The root KMS master key distributor is a peer-to-peer infrastructure running concurrently in RAM globally on dedicated machines; each gets its key material from other running instances.
    - If all instances of the distributor were to go down (total shutdown), a master key is stored in (different) secure hardware in (physical) safes in limited Google locations.
    - The root KMS master key distributor is currently being phased in, to replace a system that operated in a similar manner but was not peer to peer.

Global availability and replication

High availability and low latency, global access to keys, are critical at every level; these characteristics are needed for key management services to be used across Google.

For this reason, KMS is highly scalable, and it is replicated thousands of times in Google's data centers globally. It is run on regular machines in Google's production fleet, and instances of KMS run globally to support Google Cloud operations. As a result, the latency of any single key operation is very low.

Root KMS is run on several machines dedicated to security operations, in each data center. The root KMS master key distributor is run on these same machines, one-to-one with Root KMS. The root KMS master key distributor provides a distribution mechanism via a gossiping protocol—at a fixed time interval, each instance of the distributor picks a random other instance to compare its keys with and reconciles any differences in key versions. With this model, there is no central node that all of Google's infrastructure depends on; this allows Google to maintain and protect key material with high availability.

Google's common cryptographic library

Google's common cryptographic library is Tink,[7] which incorporates our FIPS 140-2 validated module, BoringCrypto.

Tink is available to all Google developers. Consistent use of a common library means that only a small team of cryptographers needs to implement this tightly controlled and reviewed code—it's not necessary for every team at Google to "roll their own" cryptography. A special Google security team is responsible for maintaining this common cryptographic library for all products.

The Tink encryption library supports a wide variety of encryption key types and modes, and these are reviewed regularly to ensure they are current with the latest attacks.

At the time of this document's publication, Google uses the following encryption algorithms for encryption at rest for DEKs and KEKs. These are subject to change as we continue to improve our capabilities and security.

Cryptographic primitive	**Preferred protocols**	**Other supported protocols**[8]
Symmetric encryption	• AES-GCM (256 bits)	• AES-CBC and AES-CTR (128 and 256 bits) • AES-EAX (128 and 256 bits)
Symmetric signatures (where used with AES-CBC and AES-CTR above for authentication)	• HMAC-SHA256	• HMAC-SHA512 • HMAC-SHA1

[7] Google also uses two libraries: Keymaster and CrunchyCrypt. Keymaster shares newer cryptography code in common with Tink, but uses a different key-versioning implementation and supports a wider variety of older algorithms. CrunchyCrypt is being merged with Tink.
[8] Other cryptographic protocols exist in the library and were historically supported, but this list covers the primary uses in Google Cloud.

Granularity of encryption in each Google Cloud product

Each Google Cloud service splits data at a different level of granularity for encryption.

	Google Cloud service	**Granularity of customer data encryption**[9] (size of data encrypted with a single DEK)
Storage	Cloud Bigtable	Per data chunk (several per table)
	Datastore	Per data chunk[10]
	Firestore	Per data chunk[11]
	Cloud Spanner	Per data chunk (several per table)
	Cloud SQL	• Second generation: Per instance, as in Compute Engine (each instance could contain multiple databases) • First generation: Per instance
	Cloud Storage	Per data chunk (typically 256KB-8MB)
Compute	App Engine[12]	Per data chunk[13]
	Cloud Functions[14]	Per data chunk[15]
	Compute Engine	• Several per disk • Per snapshot group, with individual snapshot ranges derived from the snapshot group master key • Per image

[9] Refers to granularity of encryption for customer content. This does not include customer metadata, such as resource names. In some services, all metadata is encrypted with a single DEK.
[10] Not unique to a single customer.
[11] Not unique to a single customer.
[12] Includes application code and application settings. Data used in App Engine is stored in Datastore, Cloud SQL, or Cloud Storage depending on customer configurations.
[13] Not unique to a single customer.
[14] Includes function code, settings and event data. Event data is stored in Pub/Sub.
[15] Not unique to a single customer.

	Google Kubernetes Engine	Several per disk, for persistent disks
	Container Registry	Stored in Cloud Storage, per data chunk
Big data	BigQuery	Several per dataset
	Dataflow	Stored in Cloud Storage, per data chunk
	Datalab	Stored in Cloud Bigtable, per notebook file
	Dataproc	Stored in Cloud Storage, per data chunk
	Pub/Sub	Rotated every 30 days[16]

[16] Not unique to a single customer.

Additional encryption options for Cloud customers

In addition to providing encryption by default in Google Cloud, we are working to offer customers additional encryption and key management options for greater control.

We want to enable Google Cloud customers to:

- Remain the ultimate custodian of their data and be able to control access to and use of that data at the finest level of granularity, to ensure both data security and privacy
- Manage encryption for their cloud-hosted data in the same way they currently do on-premises—or, ideally, better
- Have a provable and auditable root of trust over their resources
- Be able to further separate and segregate their data, beyond the use of ACLs

Customers can use existing encryption keys that they manage with Google Cloud, using the customer-supplied encryption keys feature. This feature is available for Cloud Storage and for Compute Engine for storage layer encryption.

Customers can also manage their own encryption keys on Google Cloud using Cloud Key Management Service. This product allows for application-layer encryption managed by the customer.

Research and innovation in cryptography

To keep pace with the evolution of encryption, Google has a team of world-class security engineers tasked with following, developing, and improving encryption technology. Our engineers take part in standardization processes and in maintaining widely used encryption software. We regularly publish our research in the field of encryption so that everyone in the industry—including the general public—can benefit from our knowledge. For example, in 2014 we revealed a significant vulnerability in SSL 3.0 encryption (known as POODLE), and in 2015 we identified a high-risk vulnerability in OpenSSL.

Google plans to remain the industry leader in encryption for cloud services. In terms of developing, implementing, and researching newer cryptographic techniques, we have teams working on:

- Partially homomorphic cryptography, which allows some operations to be performed on data while it is encrypted, so the cloud never sees the data in plaintext, even in memory. One place this technology is being used is as part of our experimental encrypted BigQuery client, which is openly available.
- Format- and order-preserving cryptography, which allows some comparison and ranking operations to be performed on data while it is encrypted.
- Post-quantum cryptography, which allows us to replace existing crypto primitives that are vulnerable to efficient quantum attacks with post-quantum candidates that are believed to be more robust against such attacks. The primary focus here is in researching and prototyping lattice-based public-key cryptography, including NIST recommendations on post-quantum algorithms. Lattice-based crypto is currently thought to be one of the most likely encryption techniques to be used in a post-quantum world, although we are still in early days in terms of best algorithms, concrete parameters, and cryptanalysis for applying lattice-based crypto. Although symmetric key encryption and MACs are weakened by known quantum algorithms, they can still be upgraded to similar bits of security in a post-quantum world by doubling key sizes.

Further references

Google Cloud security

For general information on Google Cloud security, see the Security section of the Google Cloud website.

Google Cloud compliance

For information on Google Cloud compliance and compliance certifications, see the Compliance section of the Google Cloud website, which includes Google's public SOC3 audit report.

G Suite security

For information on G Suite encryption and key management, see the G Suite encryption whitepaper. That whitepaper covers much of the same content included here, but focuses solely on G Suite. For all G Suite solutions, we strive to keep customer data protected, and to be as transparent as possible about how we secure it. Further information on general G Suite security is available in the G Suite Security and Compliance whitepaper.