This page discusses customer-supplied encryption keys. For other encryption options, see Data Encryption Options (/storage/docs/encryption). For examples of using this feature, see Using Customer-Supplied Encryption Keys (/storage/docs/encryption/using-customer-supplied-keys).

As an additional layer on top of Google-managed encryption keys (/storage/docs/encryption/default-keys), you can choose to provide your own AES-256 key, encoded in standard Base64 (https://tools.ietf.org/html/rfc4648#section-4). This key is known as a *customer-supplied encryption key*. If you provide a customer-supplied encryption key, Cloud Storage does not permanently store your key on Google's servers or otherwise manage your key. Instead, you provide your key for each Cloud Storage operation, and your key is purged from Google's servers after the operation is complete. Cloud Storage stores only a cryptographic hash of the key so that future requests can be validated against the hash. Your key cannot be recovered from this hash, and the hash cannot be used to decrypt your data.

When you apply a customer-supplied encryption key to an object, Cloud Storage uses the key when encrypting:

- The object's data.
- The object's CRC32C checksum.
- The object's MD5 hash.

Cloud Storage uses standard server-side keys (/storage/docs/encryption/default-keys) to encrypt the remaining metadata (/storage/docs/metadata) for the object, including the object's name. This allows you to read and update general metadata, as well as list and delete objects, without needing the customer-supplied encryption key. However, to perform any of these actions, you must have sufficient permission (/storage/docs/access-control/index) to do so.

For example, if an object is encrypted with a customer-supplied encryption key, the key must be used to perform operations on the object such as downloading or moving it. If you attempt to read the object's metadata without providing the key, you receive metadata such as the object name and `Content-Type`, but not the object's CRC32C checksum or MD5 hash. If you do supply your key with

the request for the object metadata, the object's CRC32C checksum and MD5 hash are included with the metadata.

Because most metadata can be read regardless of encryption method, do not include sensitive information in your o
ucket metadata or names.

To protect your data as it travels over the Internet during read and write operations, use Transport Layer Security, commonly known as TLS or HTTPS. TLS is required when you provide an encryption key. If you accidentally use your encryption key over an unencrypted (HTTP) connection, it is possible for an attacker to intercept your key. Because of this possibility, the Cloud Storage API returns an error message warning you that your key may be compromised. If this occurs, you should immediately rotate your keys (/storage/docs/encryption/using-customer-supplied-keys#rotating).

The following restrictions apply when using customer-supplied encryption keys:

- Cloud Storage Transfer Service (/storage-transfer/), Cloud Dataflow (/dataflow/), and Cloud Dataproc (/dataproc/) do not currently support objects encrypted with customer-supplied encryption keys.
- You cannot use the Google Cloud Console to download objects that are encrypted with a customer-**supplied** encryption key. Similarly, when you use the Google Cloud Console to upload an object, you cannot encrypt the object with a customer-**supplied** encryption key. You can perform these actions with customer-**managed** encryption keys.
- To use customer-supplied encryption keys with gsutil, you must have gsutil 4.18 or later (/storage/docs/gsutil_install).
- You can only set customer-supplied encryption keys on individual objects. You cannot set a default customer-supplied encryption key for a bucket.
- If you are performing a `compose` operation on objects encrypted by customer-supplied encryption keys, the component objects must be encrypted by the same key, and you need to provide the key with the compose request. The resulting composite object is encrypted by the same key.

There are multiple ways that you can use customer-supplied encryption keys with Cloud Storage. These include:

- Partner companies.
- The JSON and XML REST APIs.
- The gsutil command-line tool.

There are several third-party partner options for using customer-supplied encryption keys. These partners make it easier to generate an encryption key and to associate that key with an object in Cloud Storage. Partners that can supply keys for Cloud Storage include Ionic Security and KeyNexus.

To learn more, see the Cloud Storage Partners page (/storage/partners/).

When you use a customer-supplied encryption key and work directly with the JSON (/storage/docs/json_api/) or XML (/storage/docs/xml-api/overview) API, you must provide both the AES-256 key and a SHA256 hash of the key. You should store both the AES-256 key and the SHA256 hash of the key securely. Google stores the SHA256 hash of your key in the object's metadata, where you can retrieve it later. This SHA256 hash cannot be used by Google (or anyone else) to decrypt your data. It is stored as a way to uniquely identify the AES-256 key that was used to encrypt a particular object.

Include the following HTTP headers in your JSON or XML request:

Header name	Value	Description
`x-goog-encryption-algorithm`	string	The encryption algorithm to use. You must use the value **AES256**.
`x-goog-encryption-key`	string	An RFC 4648 (https://tools.ietf.org/html/rfc4648#section-4) Base64-encoded string of your AES-256 encryption key.

Header name	Value	Description
`x-goog-encryption-key-sha256`	string	An RFC 4648 (https://tools.ietf.org/html/rfc4648#section-4) Base64-encoded string of the SHA256 hash of your encryption key.

If you are performing a rewrite (/storage/docs/json_api/v1/objects/rewrite) operation with the JSON API, the headers listed above are used for encrypting the destination object, and the following headers are used for decrypting the source object:

Header name	Value	Description
`x-goog-copy-source-encryption-algorithm`	string	The encryption algorithm to use. You must use the value `AES256`.
`x-goog-copy-source-encryption-key`	string	An RFC 4648 (https://tools.ietf.org/html/rfc4648#section-4) Base64-encoded string of the source object's AES-256 encryption key.
`x-goog-copy-source-encryption-key-sha256`	string	An RFC 4648 (https://tools.ietf.org/html/rfc4648#section-4) Base64-encoded string of the SHA256 hash of the source object's encryption key.

tant: The rewrite operation makes a copy of an object. If you omit the headers listed in the first table in a rewrite oper stination object is encrypted using the default server-side encryption, not your customer-supplied encryption key.

You receive an HTTP 400 error in the following cases:

- You upload an object using a customer-supplied encryption key, and you attempt to perform another operation on the object (other than requesting or updating most metadata or deleting the object) without providing the key.
- You upload an object using a customer-supplied encryption key, and you attempt to perform another operation on the object with an incorrect key.
- You upload an object without providing a customer-supplied encryption key, and you attempt to perform another operation on the object with a customer-supplied encryption key.
- You specify an encryption algorithm, key, or SHA256 hash that is not valid.

To use a customer-supplied encryption key with gsutil (/storage/docs/gsutil), add the following option to the `[GSUtil]` section of your boto configuration file (/storage/docs/gsutil/commands/config):

Option name	Value	Description
`encryption_key`	string	An RFC 4648 (https://tools.ietf.org/html/rfc4648#section-4) Base64-encoded string of your AES-256 encryption key.

gsutil automatically calculates the SHA256 hash of your AES-256 encryption key, so there is no need to configure it ir ile.

You can optionally specify one or more decryption keys. While the `encryption_key` option is used by gsutil as both an encryption and decryption key, any `decryption_key` options you specify are used only to decrypt objects. For details, see the gsutil documentation (/storage/docs/gsutil/addlhelp/UsingEncryptionKeys).

As long as you have encryption or decryption keys in your boto configuration file, they are used for all gsutil commands. When decrypting, gsutil calculates the SHA256 hash of the supplied encryption and decryption keys and selects the correct key to use for a particular object by matching the SHA256 hash in the object's metadata.

You receive an error if you upload an object using a customer-supplied encryption key and then attempt to perform another operation on the object (other than requesting or updating metadata or deleting the object) without providing the key.

If an object is encrypted using a customer-supplied encryption key, you can rotate the object's key by rewriting the object (/storage/docs/json_api/v1/objects/rewrite). Rewrites are supported through the JSON API, but not the XML API. See Rotating an encryption key (/storage/docs/encryption/using-customer-supplied-keys#rotating) for examples of key rotation.

f your bucket uses object versioning (/storage/docs/object-versioning), be sure to delete the noncurrent versions of s after you rotate the key. Noncurrent versions are not deleted automatically and remain encrypted with the old key.