WASABI ACADEMY

Product Documentation

How secure is my data?

Wasabi is secure by default and all data stored in Wasabi hot cloud storage is always encrypted at rest (even if the data is already encrypted by the storage application prior to sending it to Wasabi). Wasabi follows industry-best security models and security design practices. Examples of Wasabi security features include:

- HTTPS is supported for the secure upload/download of data
- Buckets are only accessible to the bucket and object creators
- Wasabi supports user authentication to control access to data
- Access control mechanisms such as bucket policies and Access Control Lists (ACLs) can be used to selectively grant permissions to users and groups of users

Additional general info on Wasabi and security can be found here.

Additional specific info on Wasabi's encryption at rest methods (also known as DARE or data-at-rest-encryption) is provided below.

Wasabi system software always encrypts object data before it is written to a disk drive. Encryption is done using an AES256-bit key that can be provided in two different methods:

- If the S3 client app provides an encryption key in the S3 PUT Object Data REST request (the SSE-C approach described here), that key is used to encrypt the object data before writing to disk. After the PUT Object operation is completed, the key is discarded. The S3 client app must provide the same encryption key in an S3 GET Object REST request to access the data. The Wasabi system software does not keep a copy of the encryption key and it is only stored temporarily in memory while the object is being encrypted. The working of SSE-C with Wasabi is also demonstrated in this document.
- If no key is provided by the S3 client app (meaning SSE-C is not used), then a random AES256-bit key is generated using a cryptographic random routine in Wasabi's software. A different encryption key is generated for each object stored in the system. This AES-256 bit key is stored in the meta-data secure layer of the Wasabi system (until you delete that object) and is used again for decryption when you make a GET call for your object(s), and that way you get the same data back in your native format.