Double Encryption Proposal

Problems to address:

1. Data in transit is does not have end-to-end encryption
2. Data in storage on the cloud may not be encrypted or the cloud has complete access to encrypted files or encryption keys

Proposed Solution:

Data will be encrypted on the client-side with a data encryption key (DEK). The DEK will be encrypted with a key encryption key (KEK). The encrypted data will be sent over to the cloud. This process solves the two problems above.

When needing to access/manipulate the stored data, the decrypted DEK will be sent over to the cloud, but encrypted via a shared key. The cloud uses the DEK to decrypt the data and then the user can manipulate the data. After manipulation, the user (or cloud?) generates a new DEK to encrypt the data. The cloud should only have the DEK in memory and not to disk. If the cloud generates the DEK, then it is sent over to the client and will be encrypted with the KEK.

Alternative: Homomorphic encryption.

Protocol details:

Mutual authentication: OCSP Stapling

Data storage:

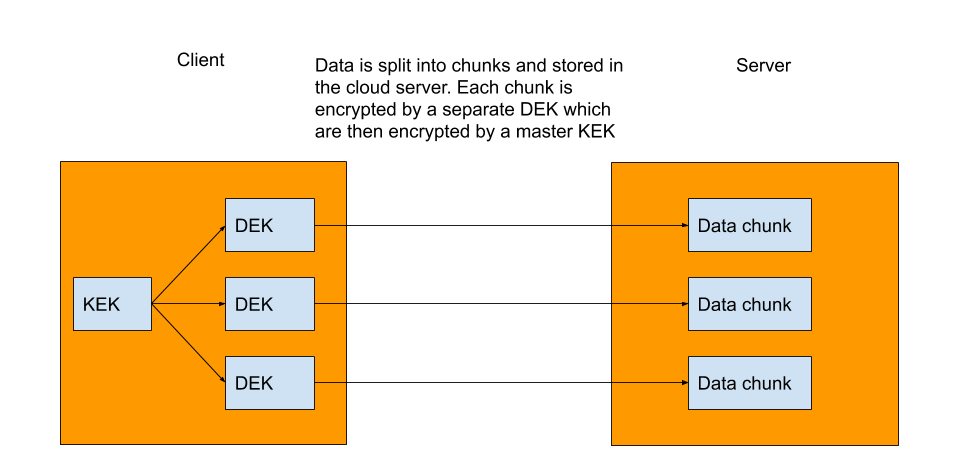
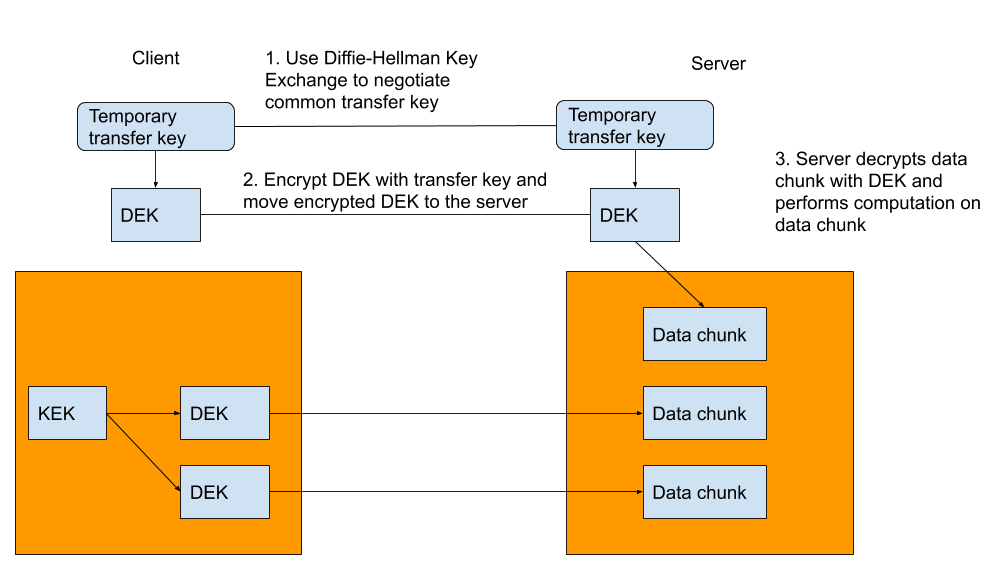
1. Data is divided into chunks at the client side.
2. Each data chunk is encrypted using a separate DEK.
3. Each encrypted data chunk is transferred to the cloud server. TLS may be used to ensure another layer of security while in-transit.
4. Each DEK is encrypted using a master KEK on the client side. Neither the DEKs and the KEK will be stored on the cloud.

Computing on the cloud:

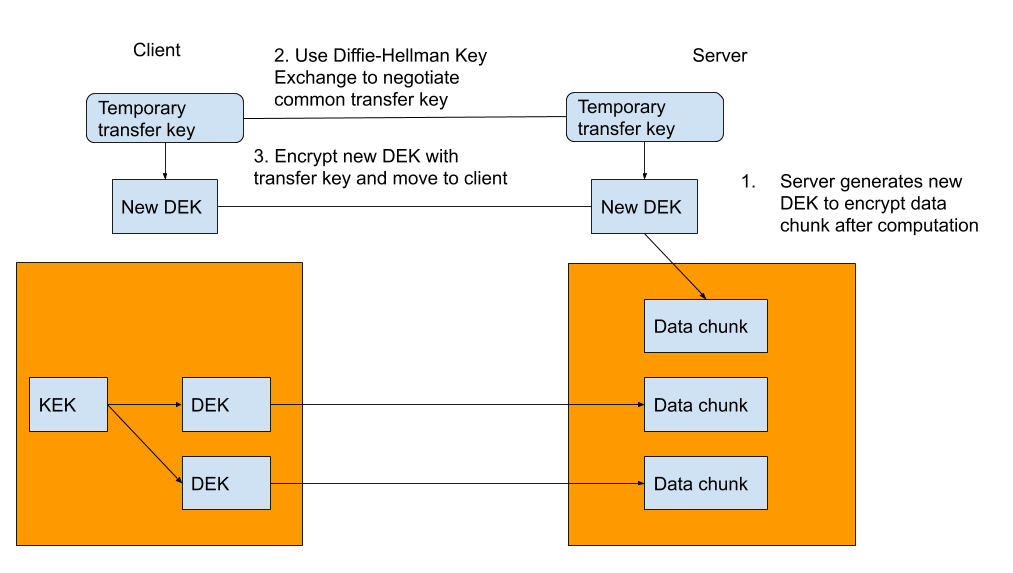
1. The client and the server identify the encrypted chunks on which computation will be done.
2. For each encrypted chunk, the client and the server negotiate a common transfer key via Diffie-Hellman key exchange. (better name for this key? Maybe key transfer key? These will only be used to securely transfer DEK between client and server)
3. The client uses the KEK to unencrypt the DEK corresponding to the chunk, and then re-encrypts it using the common transfer key.
4. The encrypted DEK is transferred to the cloud server. TLS may be used to ensure another layer of security while in-transit.
5. The cloud server decrypts the DEK using the common transfer key.
6. The cloud server decrypts the data chunk using the DEK and performs computations on the chunk.
7. (server-side DEK generation) The cloud server generates a new DEK and encrypts the chunk.
8. The client and the server negotiate a new common transfer key using Diffie-Hellman key exchange.
9. The cloud server encrypts the new DEK using the new common transfer key and transfers the encrypted DEK to the client. As always, TLS may be used for another layer of security.
10. The client decrypts the DEK using the transfer key and then re-encrypts it using the KEK.
11. The server securely deletes the DEK from disk and memory.

Diagrams:

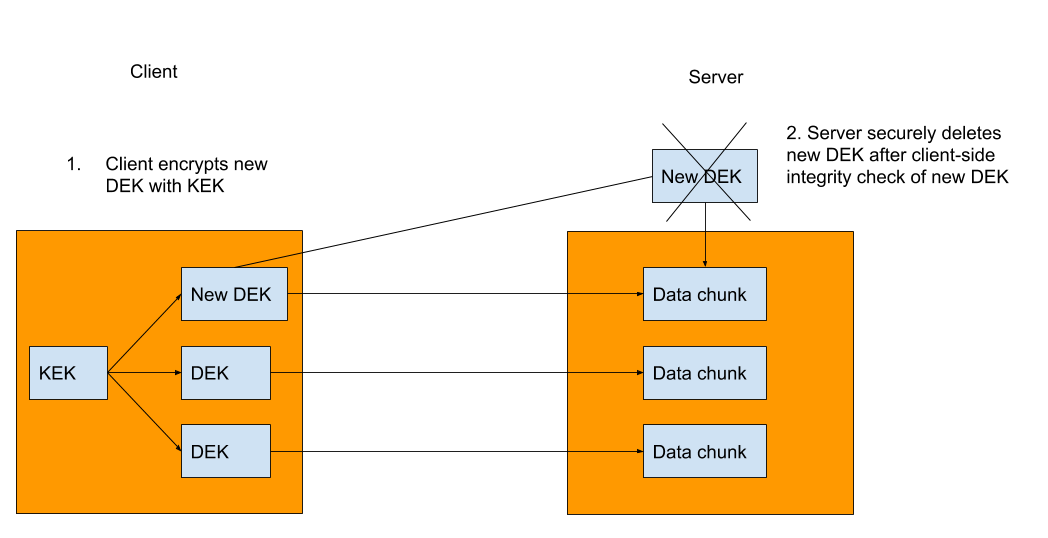
1. Data at rest

2. Decrypting data on the clouds

3.Re-encrypting data on the cloud



4. Re-encrypting DEK on client



Todo:

* Need to ensure integrity for key transfer process
  + Some kind of hashing + integrity check protocol
* Client-side key management