Concrete storage efficiency of OST-1

In this document, we describe the storage overhead incurred by OST-1 when encrypting a *single* field in a MongoDB collection with support for either of the following query types:

1. Equality
2. Range
3. None

***Remark****. This document provides* ***an estimate*** *of OST-1’s storage overhead. This is not a precise and final analysis (e.g., we don’t take into account the cost of BSON encoding, the UUID of the key)*

## Equality:

In order for our concrete analysis to make sense, we consider a user that wants to insert a 16-byte String value for an encrypted field that supports equality queries.

Inserting a single encrypted value with FLE 2.0 requires updating/inserting one document in the user’s collection, one document in the esc collection, and one document in the ecoc collection.

Therefore, the *storage overhead* can be broken down into 4 categories:

* *Ciphertext expansion (128 bytes)*: a 16 byte value results in a 128 byte ciphertext which is an 8x multiplicative factor. This is because, in addition to the value, we also store additional cryptographic objects like initialization vectors and message authentication code (MAC).
* *Tag in \_safeContent\_ (16 bytes)*: the tag is 16 bytes
* *Document in the esc collection (64 bytes)*: one document is added to the esc collection. This document has a 16-byte \_id and a 48-byte value
* *Documents in the ecoc collection (80 bytes)*: one document is added to the ecoc collection. This document has a 16-byte \_id, a 16-byte field path, and a 48-byte value.

This is a total of **288 bytes**. So the overall storage overhead is **18x** compared to a *plaintext* value.

In the case of FLE 1.0, a 16-byte value is expanded to 64 bytes when encrypted. *So FLE 2.0 has a storage overhead of* ***4.5x*** *compared to FLE 1.0*.

## Range:

We consider a user that wants to insert an int32 (4 bytes) value in a numerical field.

Inserting a single encrypted numerical value with FLE 2.0 requires updating/inserting a single document in the user’s collection, 9 documents to the esc collection, and 9 documents to the ecoc collection.

As above, the storage overhead can be broken down into 4 categories:

* *Ciphertext expansion (384 bytes)*: a 4-byte numerical value results in a 384-byte ciphertext which is a 96x multiplicative factor. This is because, in addition to the value, we also store cryptographic objects as above (plus some objects needed to support ranges).
* *Tag in \_safeContent\_ (144 bytes)*: we store 9 tags each of which is 16 bytes
* *Documents in the esc collection (576 bytes)*: we store 9 documents each of which includes a 16 byte \_id and a 48-byte value
* *Documents in the ecoc collection (720 bytes)*: we store 9 documents each of which includes a 16-byte \_id, a 16-byte field path, and a 48-byte value

This is a total of **1824 bytes**. So the overall storage overhead is **456x** compared to a plaintext value.

We can't compare this to FLE 1.0 since it does not support range queries on encrypted fields.

## None:

Inserting an encrypted value with no search/query support requires inserting/updating a single document in the user’s collection.

A 16-byte plaintext value results in a 64-byte ciphertext which is a 4x overhead and also the same overhead as FLE 1.0.