Castor EDC Encryption module

This document provides a brief overview of the functional and technical aspects of the Castor EDC encryption module.

Date: 01-08-2018
Author: D.L. Arts

Background and module overview

To give our users an additional layer of protection, we are launching our encryption module. This module allows study admins to encrypt individual Castor fields (study variables).

Because encryption keys are seperated at the institute level, this module provides a great way to restrict access to Personally Identifiable Information (PII) within a multi-center study.

Three levels of user rights granularity:

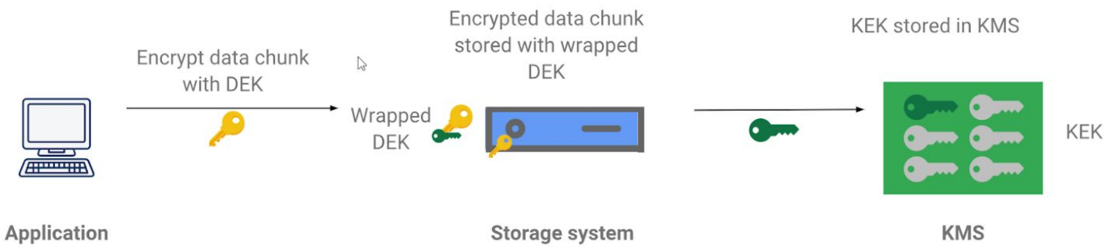
Level 1 No encryption rights	No encryption rights for any institute. This user would never be able to decrypt or encrypt any information, regardless of whether they have access to records for an institute. Typical use-case: A data manager that needs access to anonymous records of one or more institutes, to perform monitoring or to make exports for a data safety monitoring board.
Level 2 Institute specific encryption rights	Encryption rights for a single institute. A user with these rights will only be able to encrypt and decrypt data for one specific institute. Typical use-case: A medical doctor or research nurse needs to enter study data for patients that consented to study personnel seeing their full medical file including PII. These might have access to records of other institutes depending on the setup, but without encryption rights they will never be able to see PII for those records.
Level 3 Study wide encryption rights	Encryption rights for all institutes in a study. Typical use-case: A scenario which would require the patients from all hospitals to have consented to giving a study admin or data manager access to their PII and medical data. This could be useful when a central organization such as a lab has to be able to reach out to a patient to set up an appointment or when there is another central need for identifying patients.

Technical overview

Data for fields that are configured to be encrypted are encrypted locally using *"libsodium"*, an industry leading encryption library. Unencrypted data is never stored on disk and is wiped from memory as soon as it's encrypted. We have several measures in place to ensure encrypted data is never accidentally captured by logs.

We use symmetric key encryption with randomly generated keys that are unique per institute per study and Initialization Vectors for each piece of data. We generate the data encryption keys (DEKs) locally using high quality random number generators. Specifically, the algorithm to encrypt data is XSalsa20.

To protect the keys, we use a technique known as Envelope Encryption. After encrypting the data, we also then encrypt the data encryption key using Google's Key Management System and store it alongside the data locally.


Encrypt data chunk with DEK
Encrypted data chunk stored with wrapped DEK
KEK stored in KMS
Wrapped DEK
KEK
Application
Storage system
KMS

By encrypting only the key, we gain the quality of Google's encryption services but at no point does **medical data ever leave our servers or touch the Google servers**. Keys can easily be rotated on a regular basis.

In the unlikely event of a database leak, data is useless without the encryption key. However, to decrypt the encryption key, the attacker must breach Google's own key management system as well. In the meantime, we can simply rotate keys and re-encrypt the data in the background making any leaked data worthless.

For searchable encrypted medical data, we treat it with the same security as a password. We allow only exact matching search, allowing us to use a one-way hashing algorithm, Argon2i (and later Argon2id) and match only on that. Every individual field uses a unique salt requiring a potential hacker to break the encryption for each column individually for each institute. The tradeoff for making it searchable is that the same value (per field) would receive the same hash, making the search index susceptible to frequency analysis in the event of a data leak. Therefore, we recommend not making every column searchable.