Pubkey Encrypt: Architecture Document

Project Objectives:

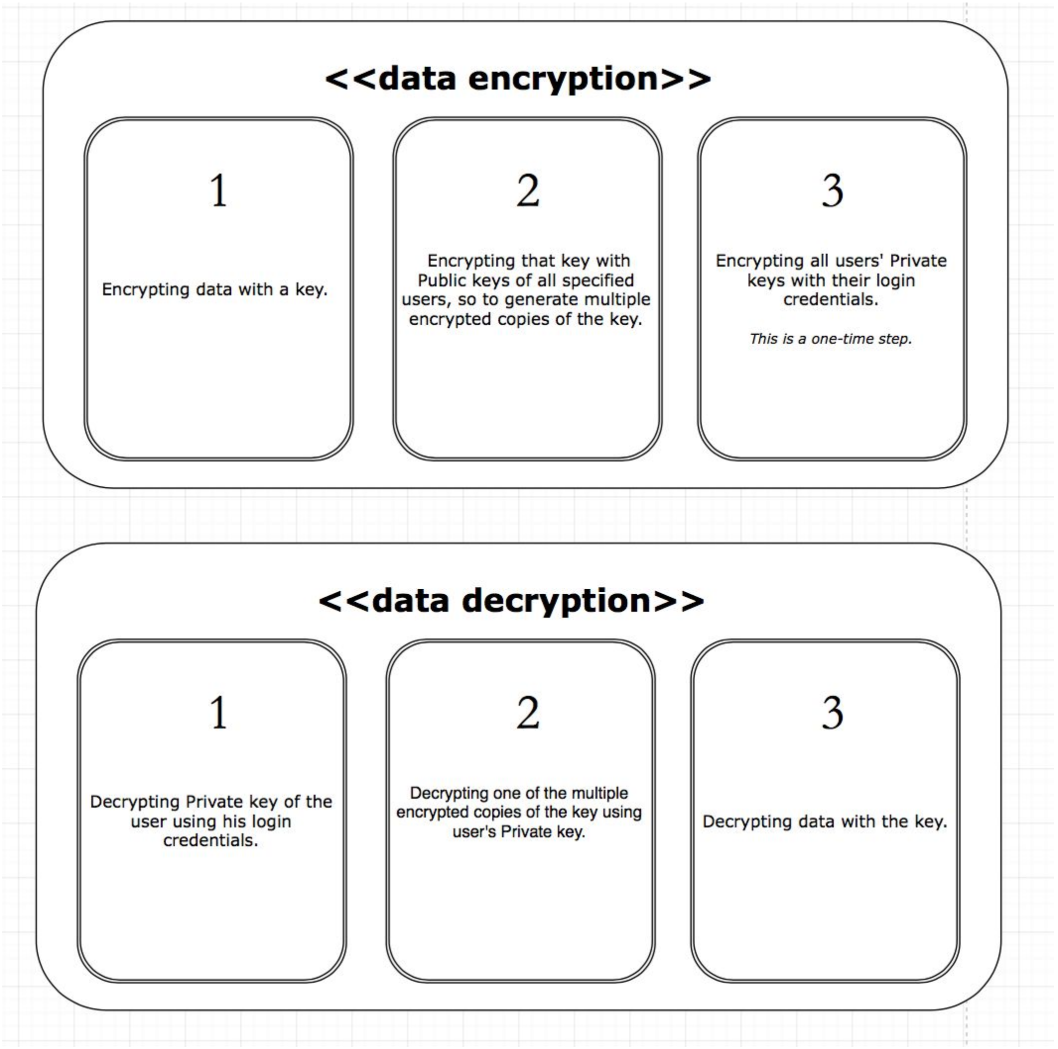
- Provide an easy-to-use website security enhancement module for administrators.
- Use ownCloud Data Encryption Model in the context of Drupal.
- Protect websites' Data at Rest in the cases of data breaches.

Project Goal:

- Add support for user credentials-based encryption mechanism into Field Encrypt.

Higher-level view of the mechanism:

The module will support a 3-step encryption/decryption mechanism based on ownCloud Data Encryption Model:


<<data encryption>>
1
Encrypting data with a key.
2
Encrypting that key with Public keys of all specified users, so to generate multiple encrypted copies of the key.
3
Encrypting all users' Private keys with their login credentials.
This is a one-time step.
<<data decryption>>
1
Decrypting Private key of the user using his login credentials.
2
Decrypting one of the multiple encrypted copies of the key using user's Private key.
3
Decrypting data with the key.

Speed, flexibility and scalability are the major benefits of using such a 3-step mechanism. Please read the ownCloud white paper if you want to know more about such non-functional attributes of this mechanism.

Project Dependencies

- Field API - allows custom fields to be attached with Drupal entities
- Key module - provides API for managing keys and key-based operations in Encrypt
- Encrypt module - provides API for managing encryption plugins and operations
- Real AES module - provides an encryption method plugin for the Encrypt module.
- Field Encrypt module - adds options to encrypt field values

How the Module will Integrate with its Dependencies:

Field API: The module will register three additional fields with the User entity

- Public key (string)
- Private key (string)
- Protected (bool)

In this way each user will get a Public/Private key pair assigned to him. The Protected bool will tell whether the Private key for a user has been saved in its original form or has it been further encrypted with that user’s credentials and then saved.

It should be noted here that the Public/Private keys assigned to users will never ever change. And the module will not allow anyone to configure these fields from the UI either.

Key API: The module will implement a Key Provider plugin such it will not store the key values in their original format. But instead:

- Upon key storage, it’ll encrypt the key value with the Public keys of all users from any role specified in the key settings (along with any users with “administer permissions” permission). Then the key provider will store these multiple encrypted copies (i.e. one encrypted copy per user) of the key in the Drupal configuration system. After that, the actual key will be discarded. Let’s call the actual key value as Role key and the multiple encrypted copies of the Role key as Share keys.
- Upon key retrieval, it’ll decrypt one of the multiple encrypted key copies with the Private key of the logged in user. Then the key provider will return the decrypted Role key.

Hence the Role keys, which will be used for actually encrypting the data, will never get stored in the server in their original format.

With this Key Provider plugin, the module will generate multiple Role keys depending upon the number of roles in a website i.e. one Role key per role.

Encrypt API: The module will generate multiple encryption profiles to be used for encrypting data. Each role defined in a website will have an associated encryption profile for it, so to allow administrators chose the most appropriate encryption profile depending upon their data.

For example, an article body visible only to people having the “Premium Users” role should be encrypted with the corresponding “Premium Users” encryption profile.

Real AES: All encryption profiles will be bundled with the Real AES encryption method.

Field Encrypt: The module will leverage the Field Encrypt module for the actual encryption/decryption events on field data.

Architecture of the Module:

The uniqueness of the module’s functionality comes from the fact that it will use user-credentials in the data encryption phases. So, the Data at Rest will remain protected even in cases of data breaches. By default, the module will add support for Password-based public key encryption which means users’ passwords will be used during both encryption and decryption of data. But it's not a far fetched assumption to say that an organization might want some other user-credentials (like PIN etc.) to be used. Therefore, we’ve decided to make:

- **A pluggable system for providing user-credentials:** The plugin would take a user ID as the input and would return the user-credentials as the output. Then it’s the duty of the module to use the provided user-credentials during data encryption and decryption phases.
- **A pluggable system for generating Public/Private key pairs.**

Here are all the events and their associated responses which the module will trigger:

- **Module Installation:** Registration of the additional three fields into the user entity.
- **Module Initialization:** Initialization of the three additional fields for all users in the website + Creation of Role keys & Encryption Profiles for all roles in the website.
- **New user registration:** Initialization of the additional three fields for a new user & Updation of Role keys for all roles of the new user.
- **User first-time login:** Protection of the Private key field with the user credentials.
- **User login:** Decrypting the Private key of the user via his credentials & temporarily storing the Private key in sessions for later accesses, if any.
- **User credentials change:** Modification of the Private key field so to protect it with new user credentials. It should also be noted here that a credentials change will not require any re-encryption of data.
- **New role addition/deletion:** Creation/deletion of the corresponding Role Key & Encryption profile.

To enhance performance, a feature of Enabling/Disabling roles will be provided in module settings form. Pubkey Encrypt would stop all its processes for the Role key of any disabled role.