wwpass®

Contemporary encryption key management architecture

The recent avalanche of data breaches attributed to successful access by unauthorized people to databases containing confidential information, suggests that we need to scrutinize the practices of storing confidential data, review so called “best practices” and identify ways to fortify security for such data.

It goes without saying that encryption of data is mandatory for preventing it from falling into the wrong hands. However, encryption in general can by no means protect the information. To be processed by legitimate users, it needs to be decrypted, and can become an easy prey while decrypted. Moreover, even while encrypted, it can still be compromised if decryption keys are not well protected. Although the pure statement that data is encrypted may formally satisfy the compliance requirements, it cannot be considered sufficient for real world data protection.

There are two major ways to improve the security of large sets of data:

1) Manage encryption keys with extremely secure methods to prevent them from falling into the wrong hands

2) Segregate the data into small chunks with separate encryption keys for each piece, so that even if a single encryption key gets compromised, the entire data set would not.

Here we look at both approaches and their practical implementation using WWPass Secure Distributed Data Storage (SDDS).

1. Protection of encryption / decryption keys

No matter what method of data encryption is used, it is necessary to keep encryption keys under strict control by authorized users. At the same time, it is often necessary to grant multiple users access to the same data, and manage access to it according to specific rules, allowing addition, suspension and removal access rights to specific users.

This secure multi-user access can be arranged by using unique symmetrical keys for encryption/decryption of the data itself, which themselves are kept in an encrypted form. These symmetric keys are encrypted by public keys of each user and stored in a dedicated key database. Before access to stored data, each user gets authenticated using the WWPass® system https://wwpass.com, and retrieves their private key from the WWPass Secure Distributed Data Storage (SDDS). This private key is used to decrypt the symmetric key, which is then used to decrypt data (Figure 1).

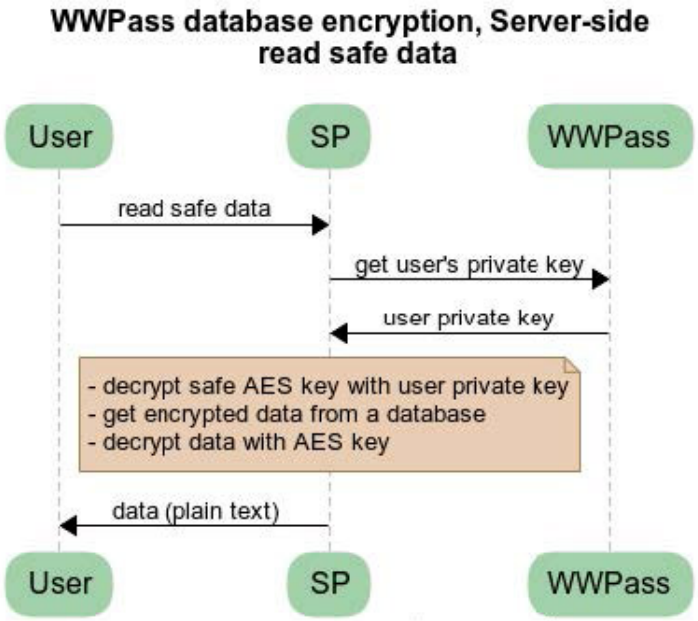

Figure 1, Read data, server-side encryption

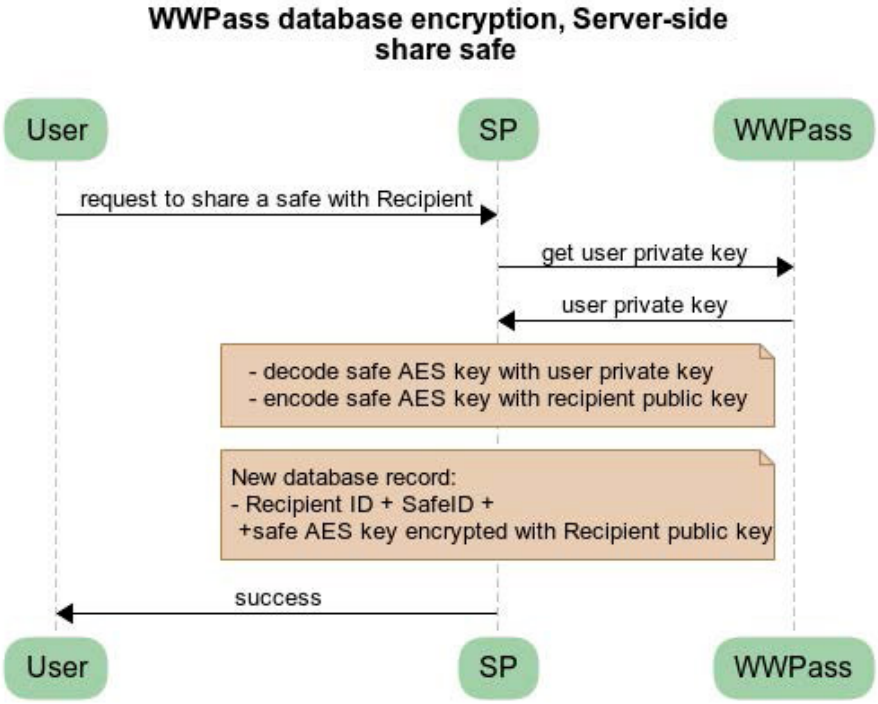

Figure 2. Share data, server-side encryption

Deleting an instance of the symmetric key, encrypted by a public key of a particular user, would disable access to the data by that particular user, without disturbing access to that data by others. Access to data can be granted to a new user through two-step "key ceremony". Initially, an authorized user which has access to a symmetric key through the ownership of its instance encrypted with their public key, sends an invitation to the other user, who in responseprovides the requestor their public key.

The first user decrypts the symmetric key with her private key, encrypts it with public key of a new user, and stores the newly created instance of the symmetric key in the key database. After that the new user can access data through use of her instance of symmetric key.

In corporate environment, if all public keys of employees are stored in corporate user database (Active Directory or LDAP, for example), the two-step "key ceremony" can be replaced with immediate access authorization. (Figure 2)

2. Data segregation and use of independent encryption keys

The bulk encryption method used in most PCI-DSS and HIPAA compliant systems, relies on a single key. This key is used to encrypt/decrypt the whole disk or database table. In reality, this approach provides virtually no protection from targeted attacks, for two major reasons:

1) Data is accessed constantly within these systems, and thus it is effectively forces system to grant access to it all the time.

2) Hackers capable of getting unauthorized access to the bulk data, usually have no trouble getting access to single encryption keys while those keys are constantly in use; so even if they steal encrypted data, they most likely also steal the single encryption key.

An alternative approach is to use unique encryption keys for each data set to be accessed by each particular user. This access does not to be one-to-one – for example, certain users may need to have access to multiple data sets, while others should only have access to specific records.

For example – a corporate employee database, where HR managers may need access to all employee records, but each employee should have access to their specific records. Such an access structure can utilize the aforementioned multiple instances of encryption keys, where each record is encrypted with its specific symmetrical key, then each symmetrical key is encrypted with public keys of each employee, and also with public key of HR manager (this must be implemented at the application level).

Additionally, these keys must not be stored in a single database (or even a few databases) as they

could be effectively stolen alongside the actual data. WWPass solves this problem by storing a critical user-specific secret on user's device (either in the WWPass mobile app or on a smartcard). This way to access data in bulk any attacker has to steal these secrets from all the user's devices, and this is enough to make untargeted attacks impractical.

Further improvement of security of data can be achieved if the encryption/decryption of data is performed on the users terminal (not on a server), and is available only during the authenticated session.

3. Client-side encryption

To dramatically reduce the risk of compromising an entire data set, client-side encryption prevents an unauthorized actor from getting access to the entire data set. It relies on a specific WWPass feature, the sitespecific "Client encryption key". Every time user logs in to particular site, a symmetric key, specific to this user/provider combination, is sent from WWPass to the browser.

When a user account is created at protected application, the browser generates asymmetric keypair, encrypts private key with "Client encryption key" and sends thus obtained public key and encrypted private key to the key database. (Figure 3)

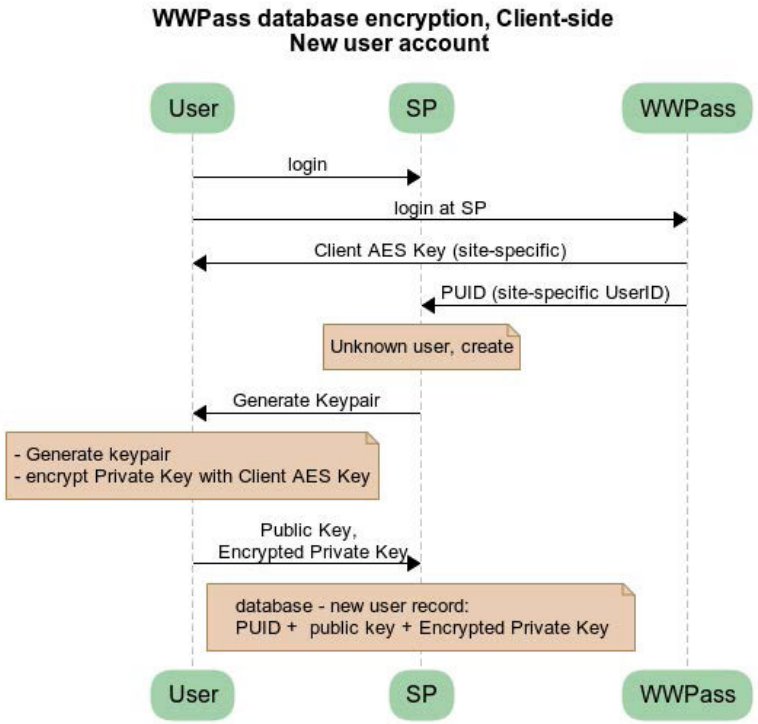

Figure 3. Creating new user account in Client-side encryption architecture

Then when the user accesses a particular record/data set, the application server cannot get it in plain text. Instead server sends all the necessary information to the browser (encrypted record data, encrypted symmetric key and user's encrypted private key). The browser obtains the "Client key" from WWPass, decrypts user private key, decrypts safe symmetric key and finally decrypts record content. (Figure 4)

Client-side encryption features ultimate security so that the application web service never gets access to any unencrypted user data.

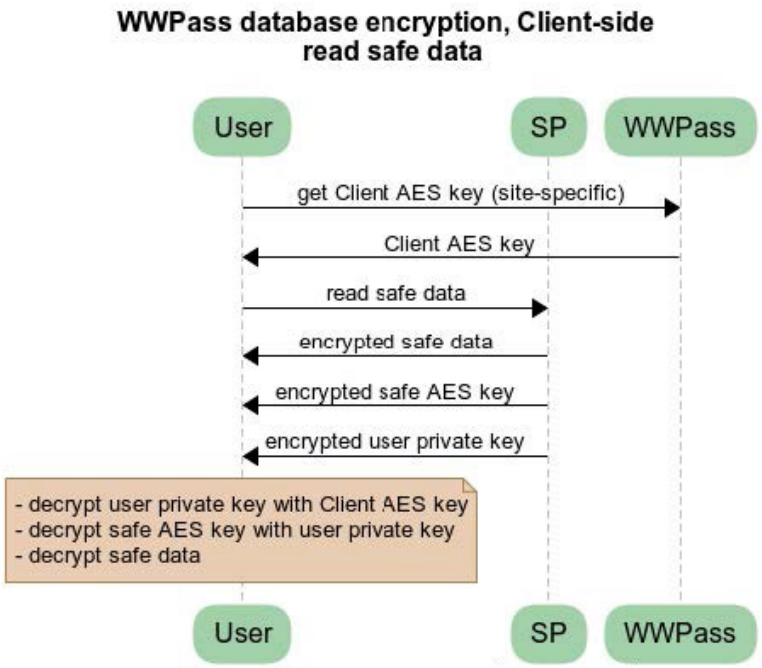

Figure 4. Reading data in Client-side encryption architecture

4. PassHub – an application which implements all these methods

PassHub™ https://passhub.net is a cloud-based password manager for individuals and teams utilizing all of the aforementioned methods to handle private information of users securely. It uses multi-user key encryption, allowing multiple secure access to shared sets of stored usernames/password (called "safes"), and client-side encryption, so nobody (including PassHub servers) ever has access to unencrypted information except the users themselves.