Card Data Flow Document

Document Reference	Card Data Flow Document
Date	17 November 2023
Document Status	FINAL
Version	1.0

Revision History

Date	User	Change Description
09-Nov-2020	Lovepreet Singh	Document Created.
23-Nov-2021	Lovepreet Singh	Azure key vault changes.
23-Oct-2022	Chinmay Jain	3DS authentication added
17-Nov-2023	Chinmay Jain	Reviewed & No Change
04-Jan-2024	Chinmay Jain	Introduction and detailed Key management overview
16-Dec-2024	Chinmay Jain	Reviewed & No Change

Contents

Introduction

This document serves as a comprehensive guide to our PCI DSS-compliant Card Vault Key Management process, designed to ensure the secure handling of sensitive cardholder data. The approach outlined here encompasses key management and rotation, card encryption, and card decryption flows, adhering to industry best practices and PCI DSS standards.

Positive Approaches:

Our key management process adopts several positive approaches to enhance security:

1. **Regular Key Rotation:** At the onset of each month, a new Data Encryption Key (DEK) is generated, promoting a proactive security stance by ensuring the continuous use of fresh encryption keys.
2. **Third-Party HSM Integration:** Leveraging the Managed Hardware Security Module (HSM) from Azure adds an extra layer of security, with no direct access to or control over the HSM, reinforcing the principle of secure key handling.
3. **External Secret Vault:** Storing the wrapped/encrypted DEK in Azure's Secret Vault, inaccessible to our infrastructure, further fortifies the security posture. The use of a unique identifier for key retrieval adds an additional layer of confidentiality.

Inclusive Diagrams:

This document includes detailed diagrams and descriptions for the following key processes:

1. **Card Encryption Flow:** Illustrating the step-by-step process of encrypting card data using the generated DEK, including key retrieval from Azure's Secret Vault and unwrapping using Azure's Managed HSM.
2. **Card Decryption Flow:** Outlining the process of decrypting card data, involving the retrieval of the DEK identifier, obtaining the wrapped/encrypted DEK from Azure's Secret Vault, and subsequent unwrapping within Azure's Managed HSM.
3. **Key Rotation Flow:** Describing the monthly generation of a new DEK, its wrapping using the KEK within Azure's HSM, and storage in the external Secret Vault.

By following this guide, our organization aims to provide a clear understanding of our secure key management practices, ensuring compliance with PCI DSS standards and fostering a robust security foundation for handling sensitive cardholder data.

Card Data Flow Diagram

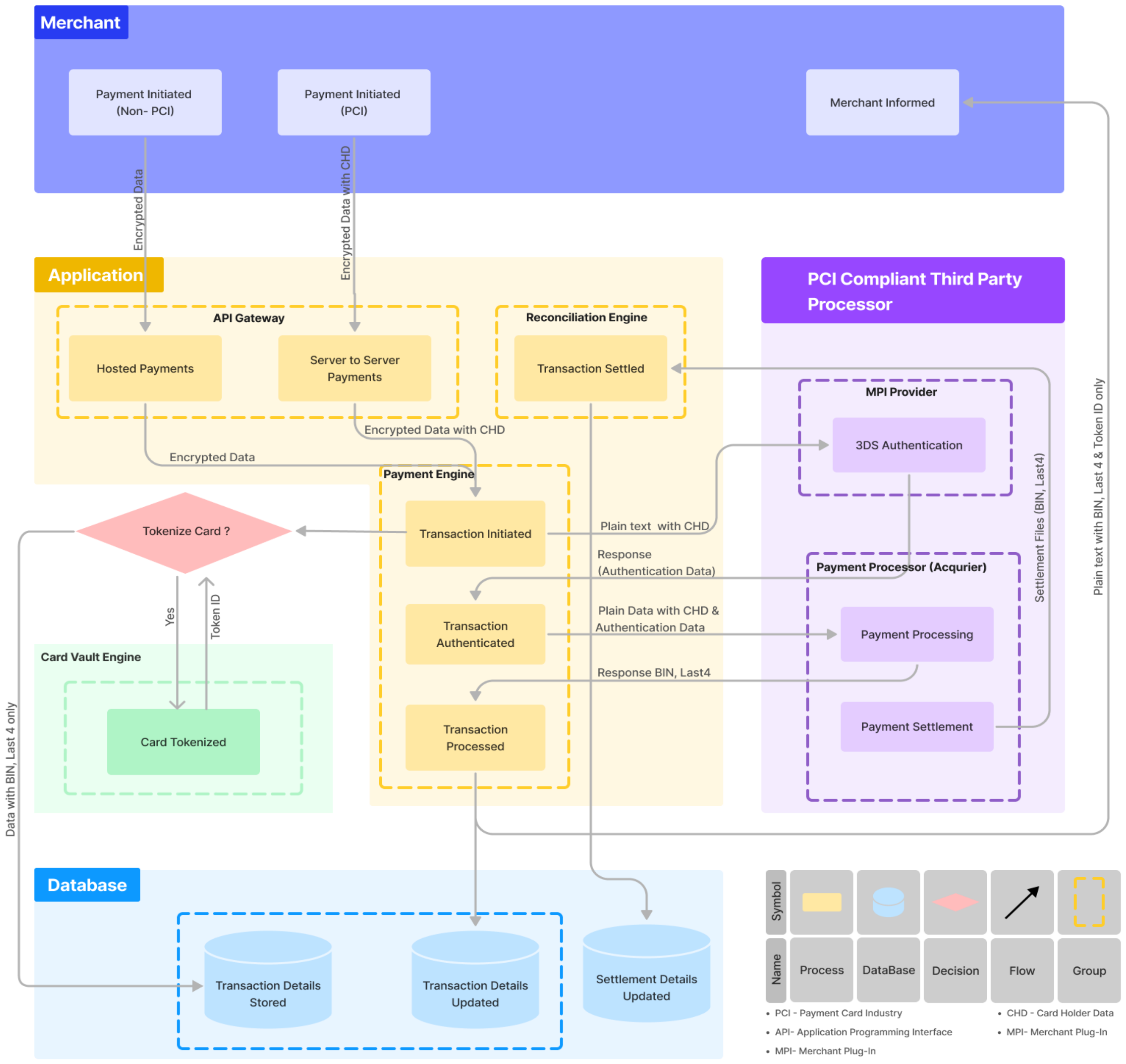


Figure Transaction Data Flow Diagram

NOTE: The step from Card Data Tokenized to Transaction Details stored in Database is achieved via the application. Card Vault is not directly connected to the database or vice versa.

Card Holder Data (CHD) Storage Details

During the complete payment lifecycle, the Cardholder data that is stored by the application is:

- Primary Account Number (PAN)
- Cardholder name
- Expiration date

The Cardholder data that is NEVER stored by the application is:

- CAV2/CVC2/CVV2/CID, or the three- or four-digit number used to verify of the card in card-not-present transactions.
- 3DS Authentication Information (PaRes, ARes, RRes, CRes)

Card Tokenization Flow

For the card tokenization, the following steps are taken by the application:

1. The unencrypted card details are collected from the customer on a PCI-compliant server.
2. The current DEK (encrypted with KEK) is retrieved from Azure Key Vault using the Secrets API
3. The encrypted DEK is sent to Azure Managed HSM for unwrapping using KEK (stored in Azure HSM).
4. The card details are encrypted using the current unwrapped DEK (AES/GCM) and then stored in the Card Vault database (cvdb). The complete list of CHD items stored in the Card Vault database are:
   4.1. Encrypted card details (AES/GCM using unwrapped DEK)
   4.2. Identifier of the current encrypted DEK
   4.3. Card Expiry Date and Card Holder Name
   4.4. Token ID generated using the Card BIN and last 4 digits concatenated with random numbers in between to ensure randomness.

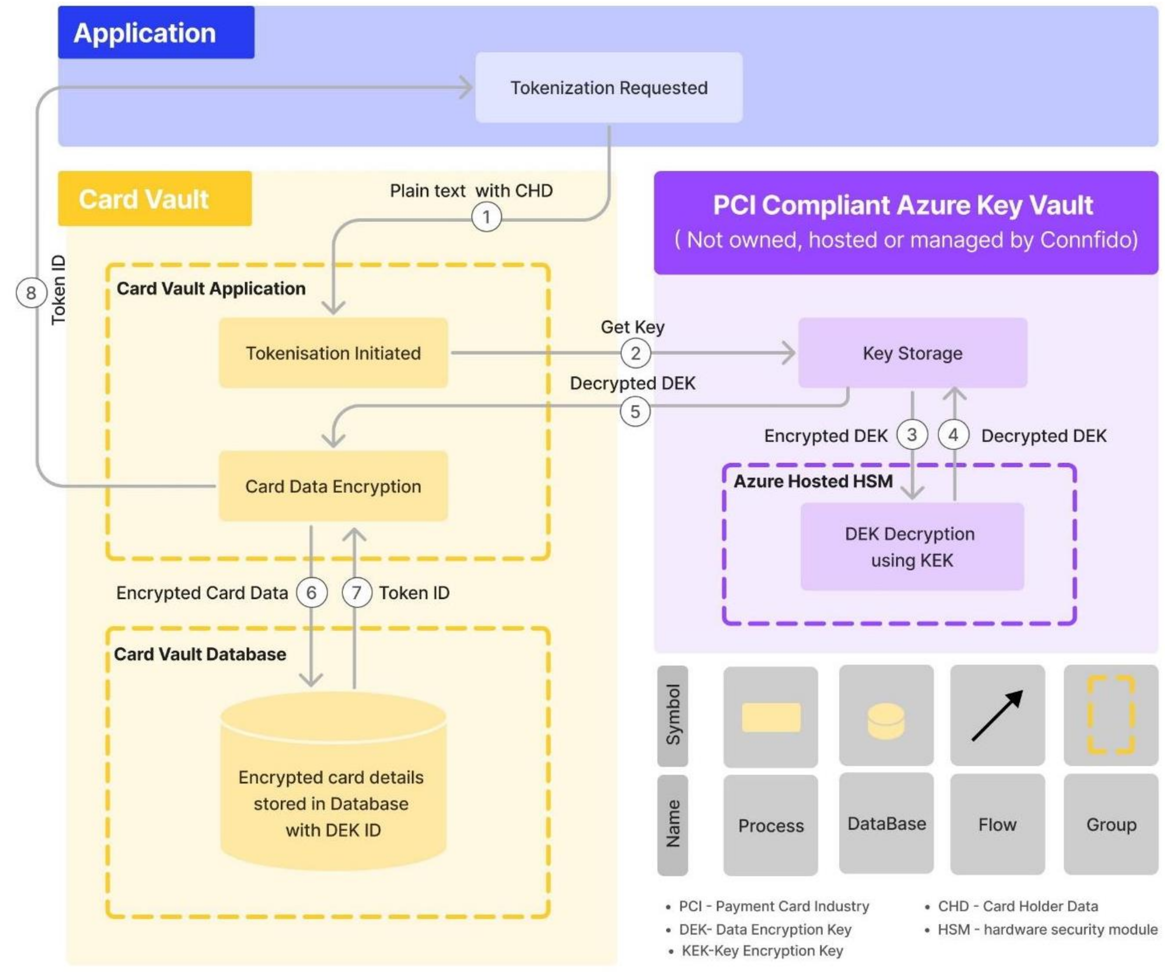


Figure 2 Card Tokenization Flow Diagram

Card Decryption Flow

For the card decryption, the following steps are taken by the application:

1. The token identifier is collected from the customer.
2. Encrypted card number and the encrypted DEK identifier is retrieved from the card vault database (cvdb)
3. The DEK (encrypted with KEK) is retrieved from Azure Key Vault via the identifier using the Secrets API.
4. The encrypted DEK is sent to Azure Managed HSM for unwrapping using KEK (stored in Azure HSM).
5. The card details are decrypted using the current unwrapped DEK (AES/GCM) and then sent back to the application.

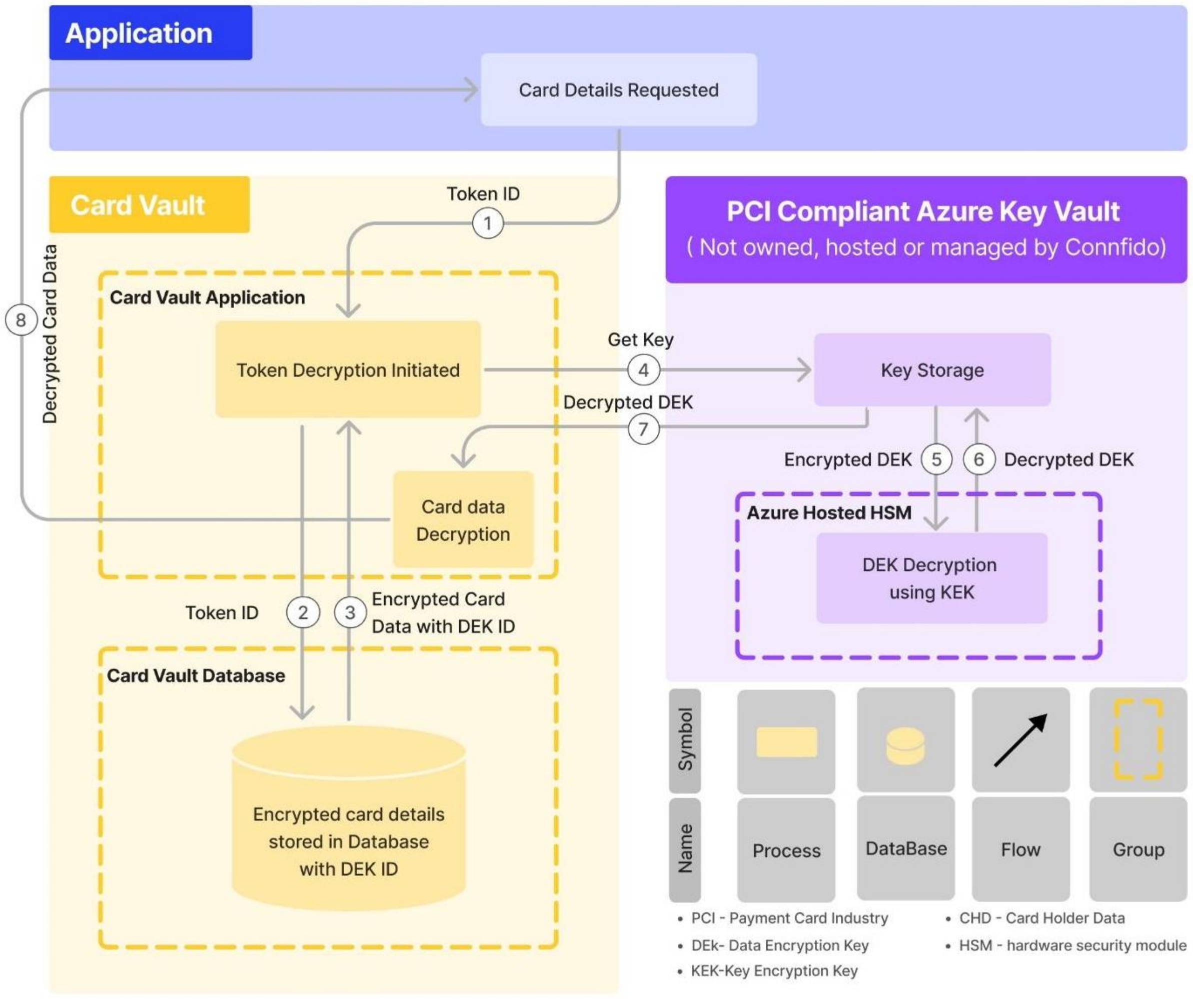


Figure 3 Card Decryption Flow Diagram

Key Management Flow

This section provides a detailed overview of the key management process for handling card data/sensitive authentication data in compliance with PCI DSS standards.

1. **Monthly DEK Generation:**
   - At the beginning of each month, our application generates a new Data Encryption Key (DEK) using the AES 256 algorithm.

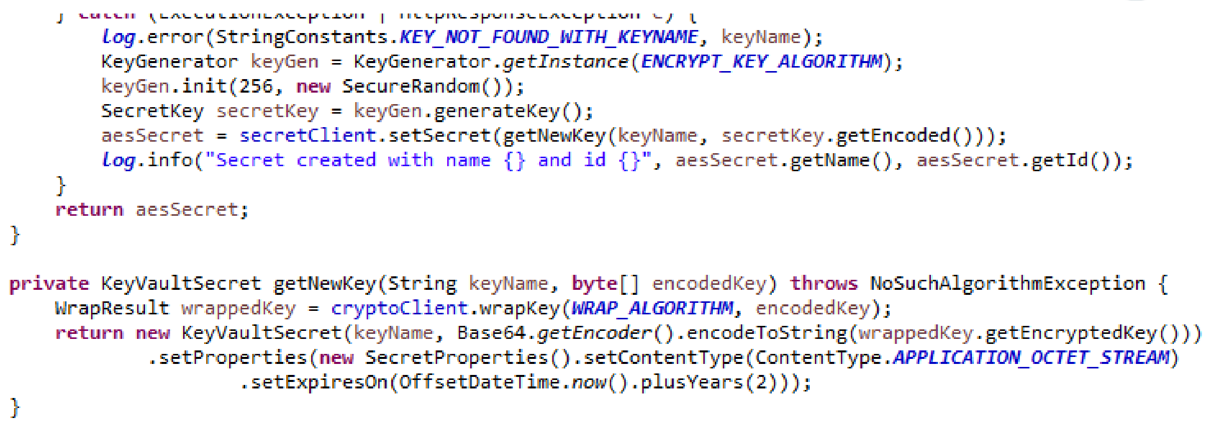
2. **Key Wrapping with Managed HSM:**
   - The newly generated DEK is sent to the Managed Hardware Security Module (HSM) of Azure for wrapping using a Key Encryption Key (KEK).
   - The wrapping process occurs exclusively within Azure's PCI-compliant HSM.

3. **Storage in External Secret Vault:**
   - The wrapped/encrypted DEK is received from Azure's HSM and stored in the Secret Vault of Azure.
   - A unique identifier is issued in response to the storage operation.

4. **Card Data Encryption/Decryption:**
   - When card data or sensitive authentication data needs to be encrypted or decrypted, the application retrieves the DEK identifier.
   - Using this identifier, a request is made to Azure's Secret Vault to obtain the wrapped/encrypted DEK.

5. **Unwrapping with Managed HSM:**
   - The obtained wrapped/encrypted DEK is sent to Azure's Managed HSM for unwrapping using the KEK.
   - The unwrapping process is conducted exclusively within Azure's PCI-compliant HSM.
   - The decrypted DEK is received in response to the unwrapping operation.

6. **Card Data Processing:**
   - The decrypted DEK is utilized for encrypting/decrypting the card data or sensitive authentication data.



```
    } catch (ExecutionException | HttpResponseException e) {
        log.error(StringConstants.KEY_NOT_FOUND_WITH_KEYNAME, keyName);
        KeyGenerator keyGen = KeyGenerator.getInstance(ENCRYPT_KEY_ALGORITHM);
        keyGen.init(256, new SecureRandom());
        SecretKey secretKey = keyGen.generateKey();
        aesSecret = secretClient.setSecret(getNewKey(keyName, secretKey.getEncoded()));
        log.info("Secret created with name {} and id {}", aesSecret.getName(), aesSecret.getId());
    }
    return aesSecret;
}

private KeyVaultSecret getNewKey(String keyName, byte[] encodedKey) throws NoSuchAlgorithmException {
    WrapResult wrappedKey = cryptoClient.wrapKey(WRAP_ALGORITHM, encodedKey);
    return new KeyVaultSecret(keyName, Base64.getEncoder().encodeToString(wrappedKey.getEncryptedKey()))
            .setProperties(new SecretProperties().setContentType(ContentType.APPLICATION_OCTET_STREAM)
                    .setExpiresOn(OffsetDateTime.now().plusYears(2)));
}
```

Figure 4 Key Generation Logic

Note:

- We do not own, host, or manage Azure's HSM.
- We do not own, host, or manage Azure's Secret Vault.
- All communication with Azure's HSM and Secret Vault is handled through Azure's SDK.
- The application has no control over the communication process, and Azure's SDK manages all interactions with Azure's infrastructure (including but not limited to getting a secret, saving a secret, wrapping, and unwrapping DEK).
- The application has no access to KEK.
- The application does not permanently store the actual DEK or encrypted DEK at any point in the process. The details are released from memory as soon as the relevant action is performed.

Key Rotation Flow

This section outlines the process of key rotation, which occurs every 3 years (with a maximum of 5 years). The key rotation process ensures the continuous security of our Data Encryption Keys (DEKs) and involves a systematic approach to update encryption keys.

1. **Eligibility Check:**
   - Every 3 years, a DEK becomes eligible for rotation.
   - The application retrieves the identifiers of all DEKs eligible for rotation.

2. **Card Data Retrieval:**
   - The application retrieves all encrypted card data and sensitive authentication data from the database that have been encrypted using the DEKs identified in step 1.

3. **DEK Unwrapping:**
   - For each set of card data, the application retrieves the wrapped/encrypted DEK from Azure's Secret Vault using the corresponding DEK identifier.
   - The wrapped/encrypted DEK is sent to Azure's Managed HSM for unwrapping.
   - Upon successful unwrapping, the application obtains the actual unwrapped DEK.

4. **Card Data Decryption:**
   - The application decrypts the card numbers and sensitive data that have been encrypted using the DEK obtained in the last step.

5. **Current DEK Retrieval:**
   - The application retrieves the wrapped/encrypted DEK for the current month from Azure's Secret Vault using the current DEK identifier.
   - The current month's wrapped/encrypted DEK is sent to Azure's Managed HSM for unwrapping.

6. **Card Data Re-encryption:**
   - Using the current DEK, the application re-encrypts the card details obtained in step 4.
   - The re-encrypted card details, along with the identifier of the current DEK, are stored in the database.

Note:

- The entire key rotation process is conducted in batches at the start of every month.
- The application does not permanently store the actual DEK or encrypted DEK at any point in the process. The details are released from memory as soon as the relevant action is performed.