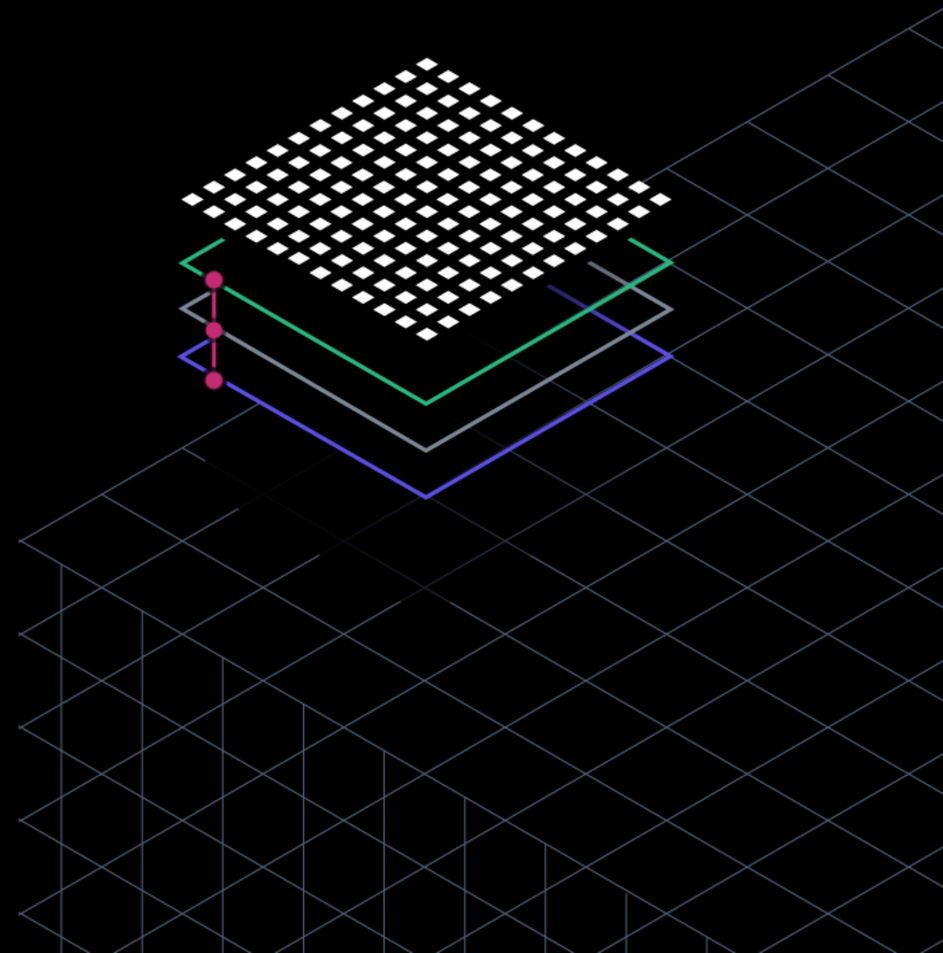
HashiCorp

Encryption as a Service



Agenda

Data encryption challenge

Encryption as a Service with Transit secrets engine

- Supported key types
- Transit secrets workflow

Encryption key rotation

- Key configuration
- Re-wrapping data

Transit Secrets Engine

Security Engineer

Encrypting data requires collaborative effort across the organization (data producer, data consumer, engineering team, etc.)

Applications must implement encryption and decryption of data

How do we manage the encryption key?

Encryption as a Service: Transit Secrets Engine

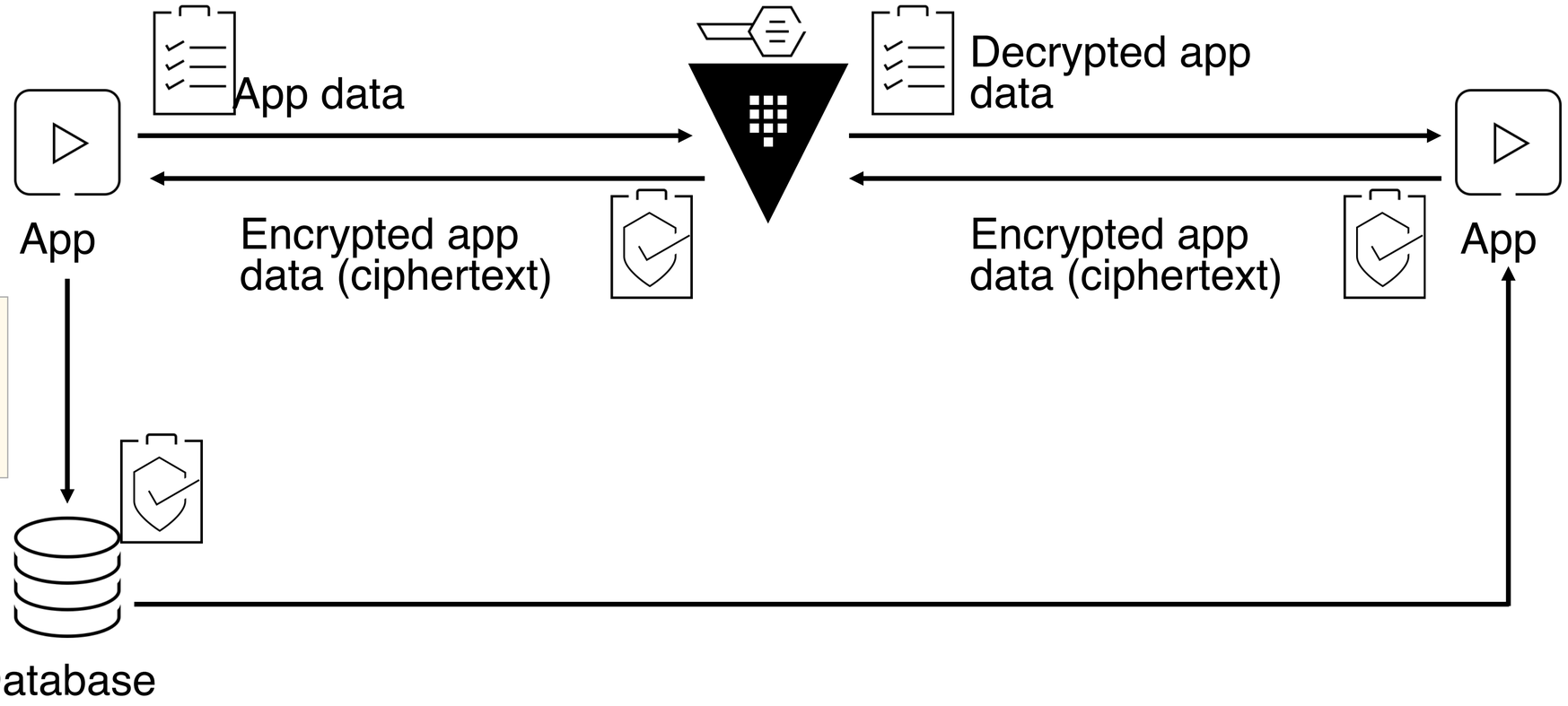
Transit secrets engine handles cryptographic functions on data in-transit

- Allow applications to *outsource* secrets management and encryption
  - Applications **never have access** to the underlying encryption keys
  - Decouples storage from encryption and access control
- Convergent encryption is a mode where the same set of plaintext+context always result in the same ciphertext and supported to allow *searchable ciphertext*
- Keys can be **rotated** and rewrap the ciphertext with new key
  - Configurable **required minimum key version** for decryption
  - Prevents old or lost ciphertext from being decrypted

Encryption as a Service

App sends data to Vault and have it encrypted

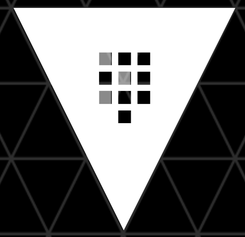
Vault decrypts the ciphertext for the app to use



Ciphertext can be stored in the primary data store

Transit Secrets Engine: Key Types

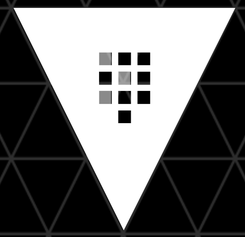
Key Type	Description
aes256-gcm96	AES-GCM with a 256-bit AES key and a 96-bit nonce (encryption, decryption, key derivation, and convergent encryption)
chacha20-poly1305	ChaCha20-Poly1305 with a 256-bit key (encryption, decryption, key derivation, and convergent encryption)
rsa-2048	2048-bit RSA key (encryption, decryption, signing, and signature verification)
rsa-4096	4096-bit RSA key (encryption, decryption, signing, and signature verification)
ed25519	Ed25519 (signing, signature verification, and key derivation)
ecdsa-p256	ECDSA using the P256 elliptic curve (signing and signature verification)



Setup transit secrets engine

Terminal

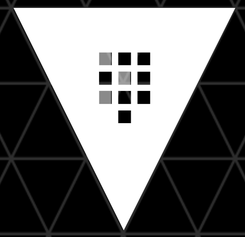
```
# Enable transit secrets engine
$ vault secrets enable transit

# Create a key to encrypt 'phone_number'
$ vault write -f transit/keys/phone_number

# Specify the key type (default would be used if not specified)
$ vault write transit/keys/ssn \
      type="chacha20-poly1305"

/* Vault HTTP API imposes a maximum request size of 32MB to prevent a denial of service attack.
This can be tuned per listener block in the Vault server configuration. */
```



Encrypt / Decrypt Data

Terminal

```
# Encrypt data with 'phone_number' key
$ vault write transit/encrypt/phone_number \
      plaintext=$(base64 <<< "1-(415)123-4567")

# Decrypt data with 'phone_number' key
$ vault write transit/decrypt/phone_number \
      ciphertext="vault:v1:tgx2vsxtlQRfyLSKvem..."
 Key           Value
 ---           -----
 plaintext     MS0oNDE1KTEyMy00NTY3Cg==

# Decode
$ base64 --decode <<< "MS0oNDE1KTEyMy00NTY3Cg=="
```



Encrypt / Decrypt Source Code

Terminal

*# Create a key to encrypt '**code_java**'*

$ vault write -f transit/keys/code_java

 # Encrypt data with 'code_java' key

$ vault write transit/encrypt/code_java plaintext=$(base64 -i TransitUtil.java)

$ vault write --field=ciphertext transit/encrypt/code_java plaintext=$(base64 -i TransitUtil.java) > TransitUtil.java_ciphertext

 # Decrypt data with 'code_java key

$ vault write transit/decrypt/code_java ciphertext=@TransitUtil.java_ciphertext

$ vault write -field=plaintext transit/decrypt/code_java ciphertext=@TransitUtil.java_ciphertext > TransitUtil.java_base64_text

 # Decode

$ base64 -i TransitUtil.java_base64_text -o TransitUtil.java_decode --decode

Encrypt / Decrypt Source Code (REST API)

Terminal

*# Create a key to encrypt '**code_java**'*

$ curl --header "X-Vault-Token: ..." --request POST --data @payload.json \ http://127.0.0.1:8200/v1/transit/keys/code_java

$ cat payload.json
{ "type": "aes256 –gcm96" }

 # Encrypt data with 'code_java' key (payload.json에 base64_encod된 소스값 지정
$ base64 -i TransitUtil.java -o TransitUtil.java_base64_encode
**$ curl --header "X-Vault-Token:…" --request POST --data @payload.json **
http://127.0.0.1:8200/v1/transit/encrypt/code_java |jq .data.ciphertext

#palyload.json {"plaintext" : "~~~~"}
 # Decrypt data with 'code_java key
$ curl --header "X-Vault-Token: ..." --request POST --data @payload.json \ http://127.0.0.1:8200/v1/transit/decrypt/code_java

#palyload.json {"ciphertext" : "~~~~"}

 # Decode
$ base64 -i TransitUtil.java_base64_text -o TransitUtil.java_decode --decode

Key Rotation

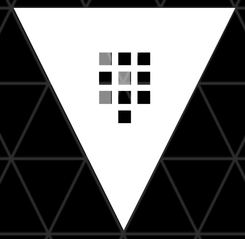
Key Rotation

Transit secrets engine allows **easy key rotation**

Keys can be rotated manually or by an automated process

Vault maintains the versioned keyring

- Admins can decide the **minimum key version allowed** for decryption operations
- You can upgrade already encrypted data to a new key



Key Rotation

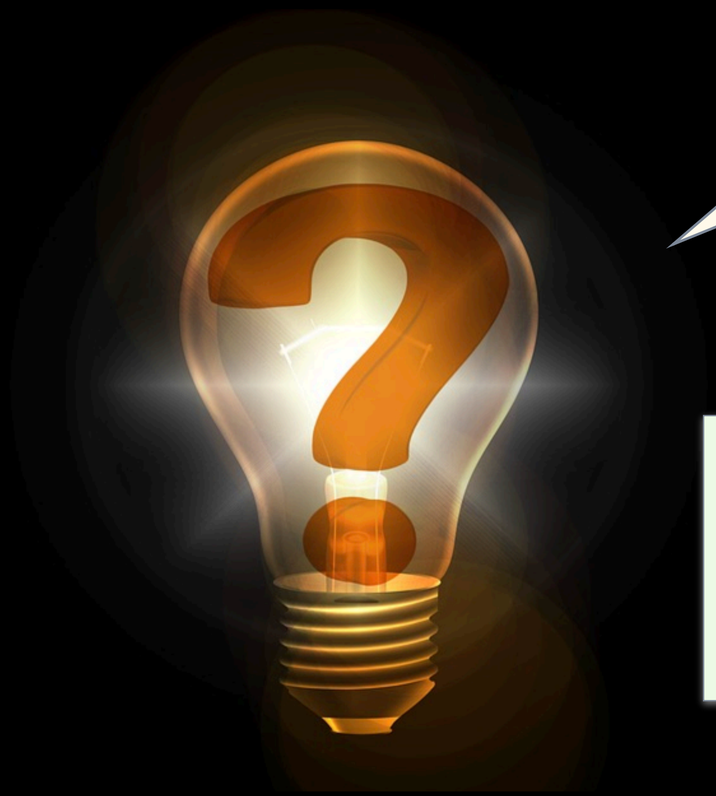
Terminal

```
# Rotate the 'phone_number' key
$ vault write -f transit/keys/phone_number/rotate

# Read 'phone_number' key
$ vault read transit/keys/phone_number
 Key                         Value
 ---                         -----
 ...
 keys                        map[1:1549341798 2:1549346826
3:1549347105 4:1549347108 5:1549347109 6:1549347110]
 latest_version              6
 min_available_version       0
 min_decryption_version      1
 min_encryption_version      0
 ...
```

Key Configuration

Terminal

```
# Update key configuration
$ vault write transit/keys/phone_number/config \
      min_decryption_version=5

$ vault read transit/keys/phone_number
 Key                         Value
 ---                         -----
 ...
 keys                        map[5:1549347109 6:1549347110]
 latest_version              6
 min_available_version       0
 min_decryption_version      5
 min_encryption_version      0
 ...
```



What happen to those ciphertext that were encrypted with v1, v2, v3, or v4 of the key?

A: Vault would refuse to decrypt the data as the key used is less than the minimum key version allowed.

This prevents old copies of ciphertext from being decrypted, should they fall into the wrong hands.

Re-wrap Data

Terminal

*# Upgrade already encrypted data to a **new key***

```
$ vault write transit/rewrap/phone_number \
      ciphertext="vault:v1:tgx2vsxtlQRfyLSKvem..."

Key            Value
---            -----
ciphertext     vault:v6:Xa1f9FIJtn13em/Wb7QCsXsU/kCOn7...
```

Thank you.

HashiCorp

www.hashicorp.com hello@hashicorp.com