Base64 is not encryption

A better story for Kubernetes secrets

@sethvargo

Developer Relations Engineer

What's a secret?

Secret (noun)

Credentials, configurations, API keys, or other pieces of information needed by an application at build time or run time

Why protect secrets?

- Attractive target for hackers
- Often leaked in repos or storage buckets
- Frequently includes overly broad permissions

Protecting secrets

Audit

Verify and log the use of individual secrets to a central system

Encrypt

Always encrypt secrets in transit with TLS and at rest

Rotate

Change a secret regularly or in case of suspected compromise

Isolate

Separate where secrets are used from where secrets are managed

Protecting secrets

Audit

Verify and log the use of individual secrets to a central system

Encrypt

Always encrypt secrets in transit with TLS and at rest

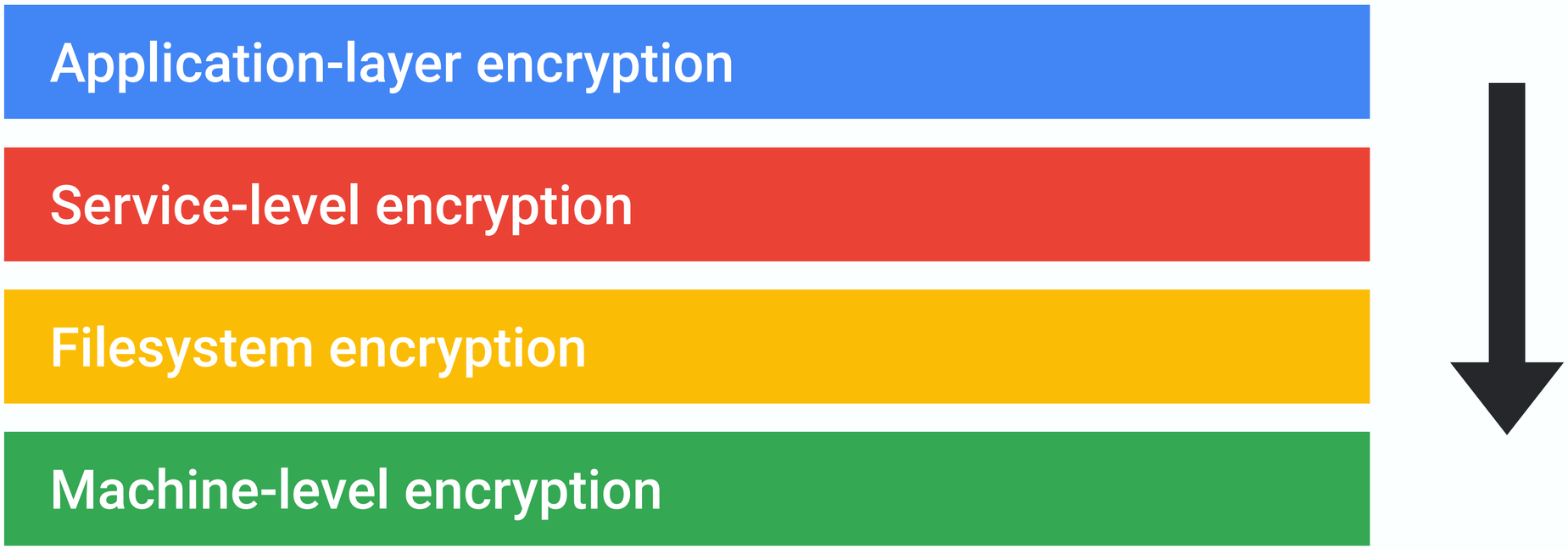
Rotate

Change a secret regularly or in case of suspected compromise

Isolate

Separate where secrets are used from where secrets are managed

Layers of encryption



App-layer encryption

- Applied at earliest possible step
- Provides protection a very granular level
- Protects data as it moves through the system

Kubernetes defaults

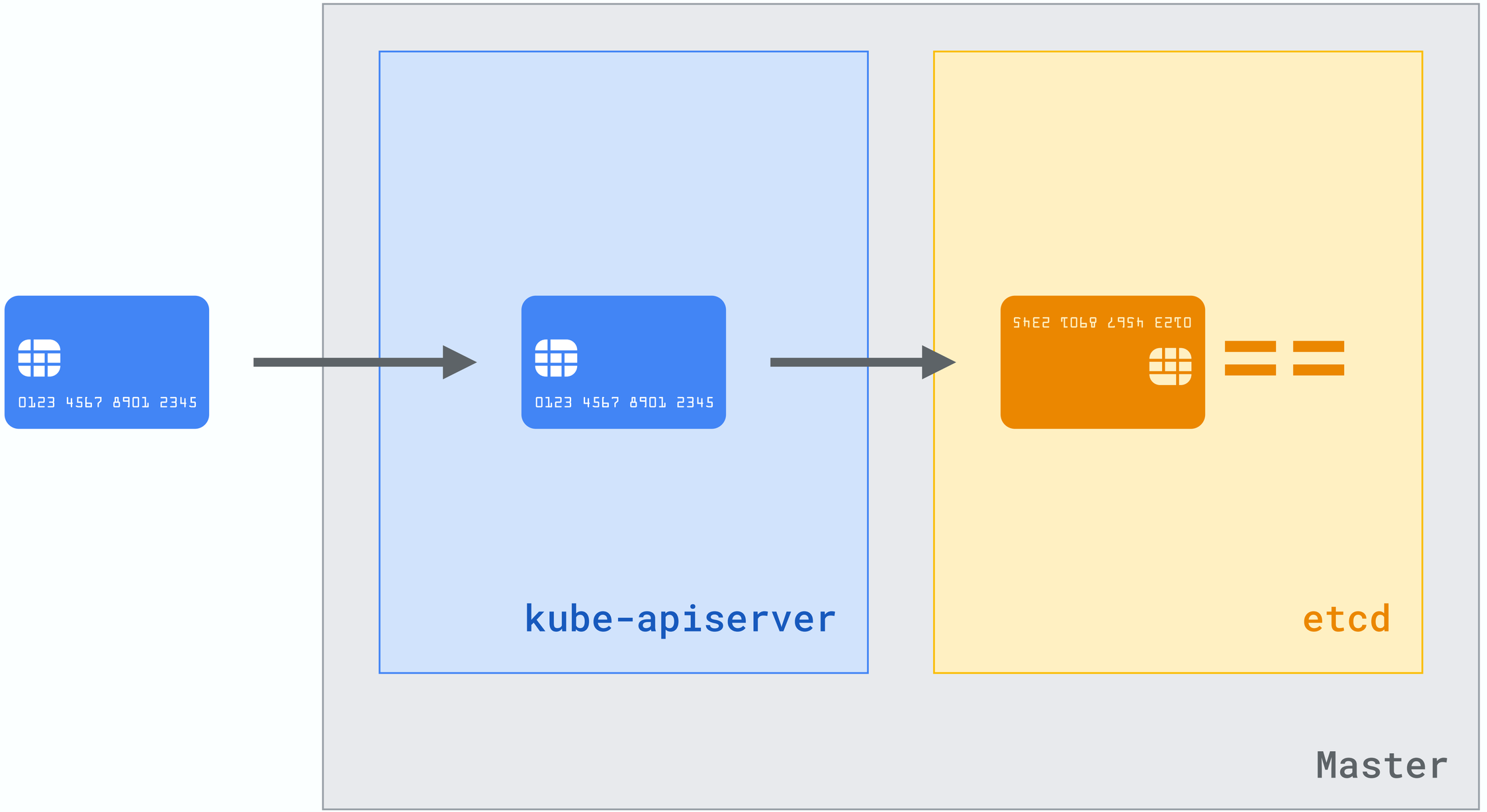
Insecure by default

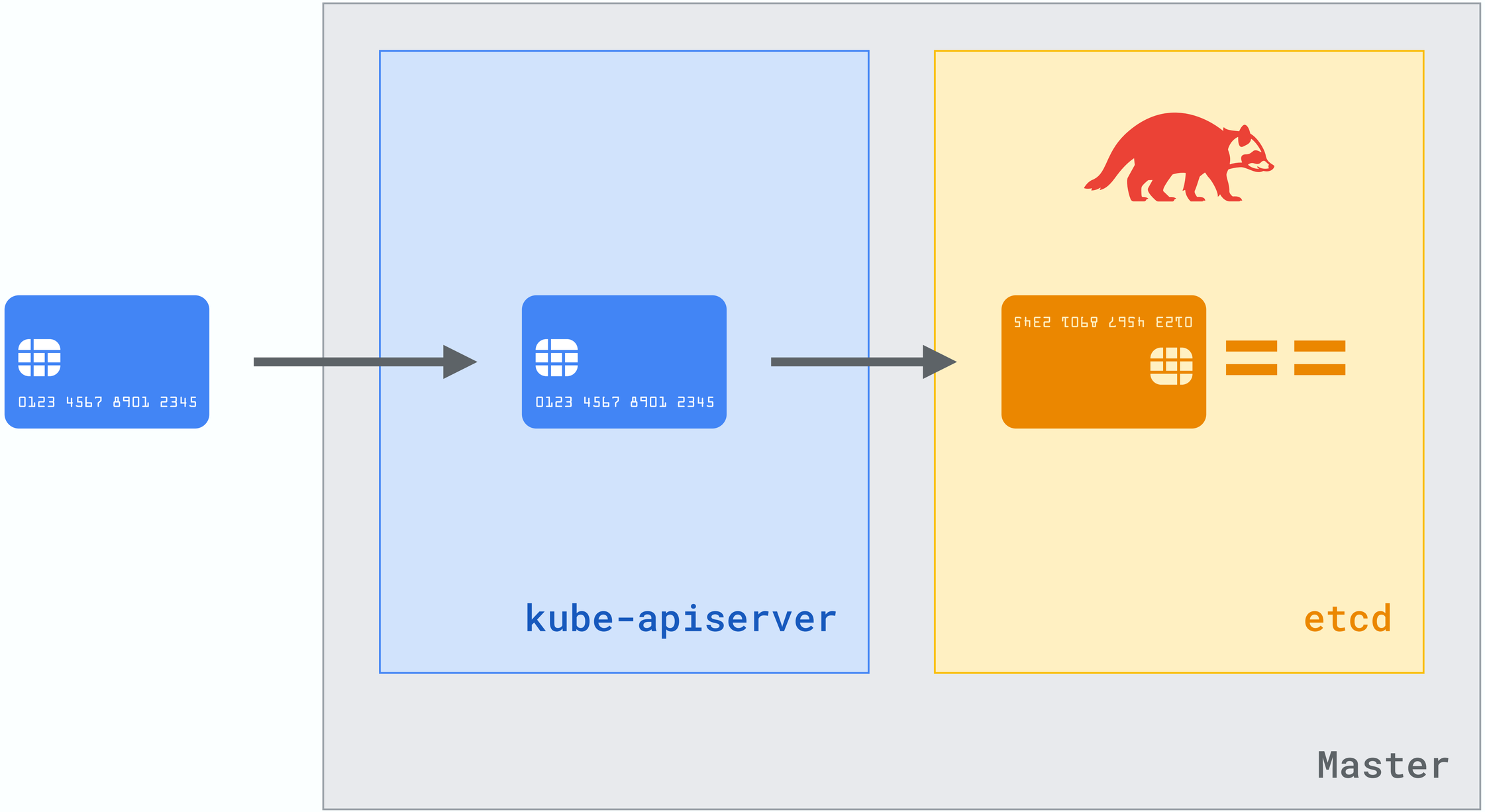
Secrets are stored in plaintext in etcd. They are base64-encoded, but not encrypted.

Insecure by default*

Secrets are stored in plaintext in etcd. They are base64-encoded, but not encrypted.

* Many providers alter this default behavior.

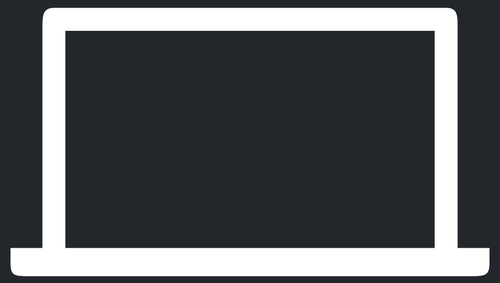

0123 4567 8901 2345
0123 4567 8901 2345
kube-apiserver
etcd
Master


0123 4567 8901 2345
0123 4567 8901 2345
kube-apiserver
etcd
Master

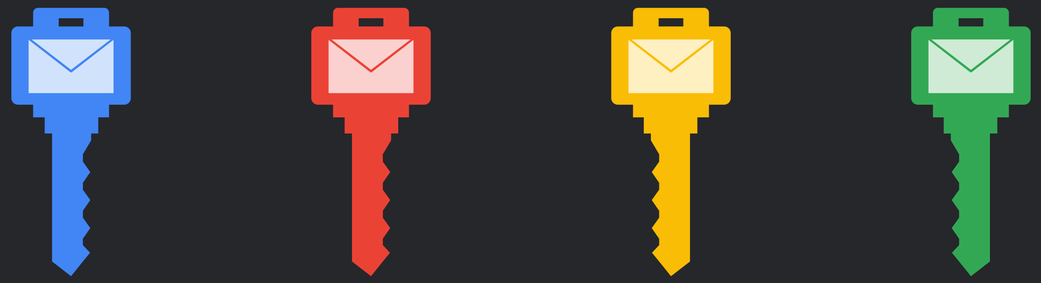
Encraption

shodan.io/search?query=etcd

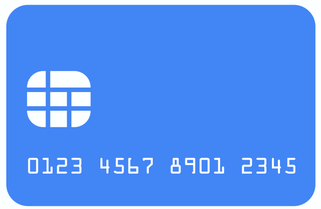
Demo



Envelope encryption



Envelope encryption



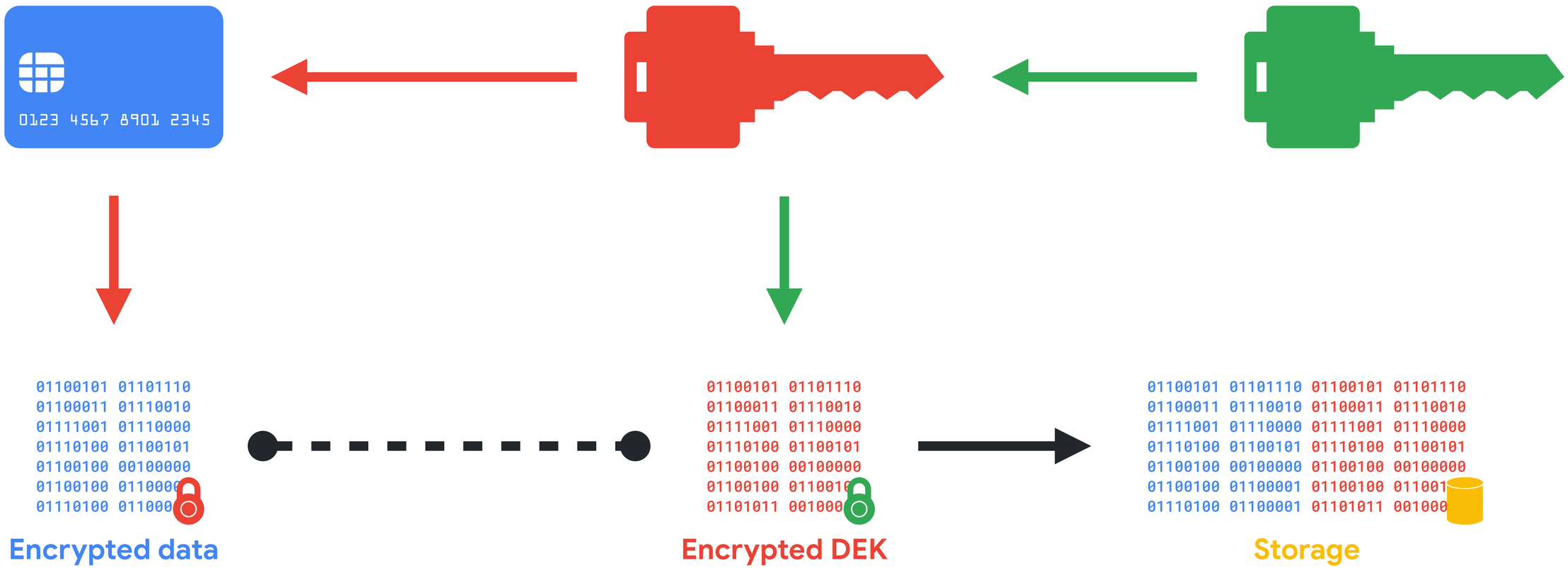
Data

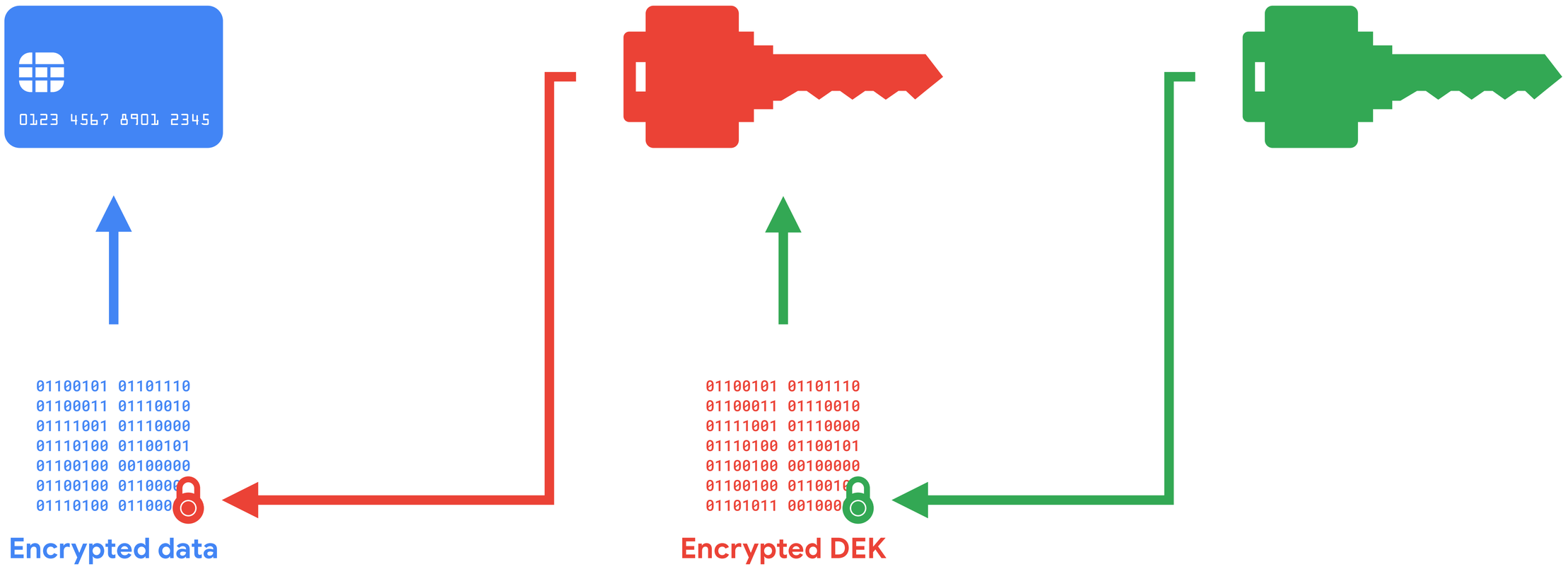
DEK

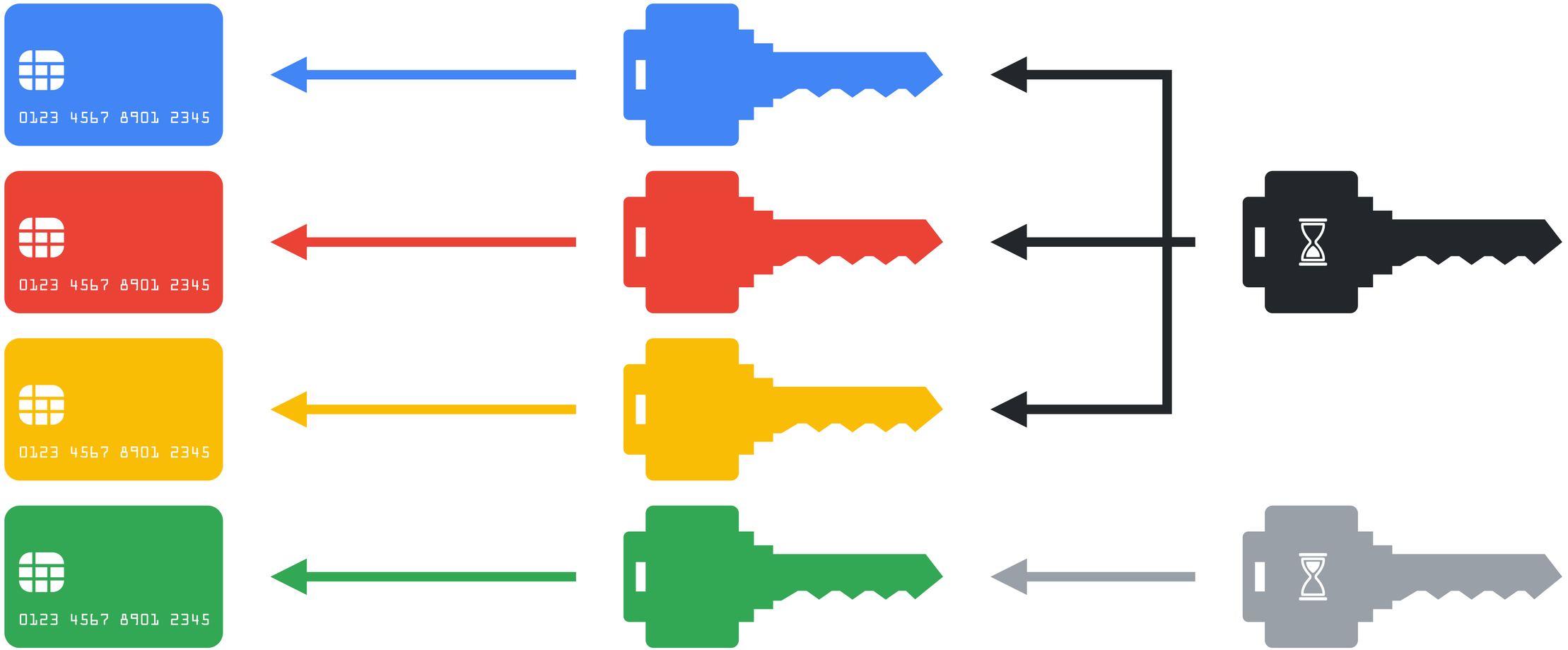
Data encryption key

KEK

Key encryption key


0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 011000
01110100 011000
Encrypted data
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 011001
01101011 001000
Encrypted DEK
01100101 01101110 01100101 01101110
01100011 01110010 01100011 01110010
01111001 01110000 01111001 01110000
01110100 01100101 01110100 01100101
01100100 00100000 01100100 00100000
01100100 01100001 01100100 011001
01110100 01100001 01101011 001000
Storage


0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 011000
01110100 011000
Encrypted data
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 011001
01101011 001000
Encrypted DEK


0123 4567 8901 2345
0123 4567 8901 2345
0123 4567 8901 2345
0123 4567 8901 2345

Envelope encryption

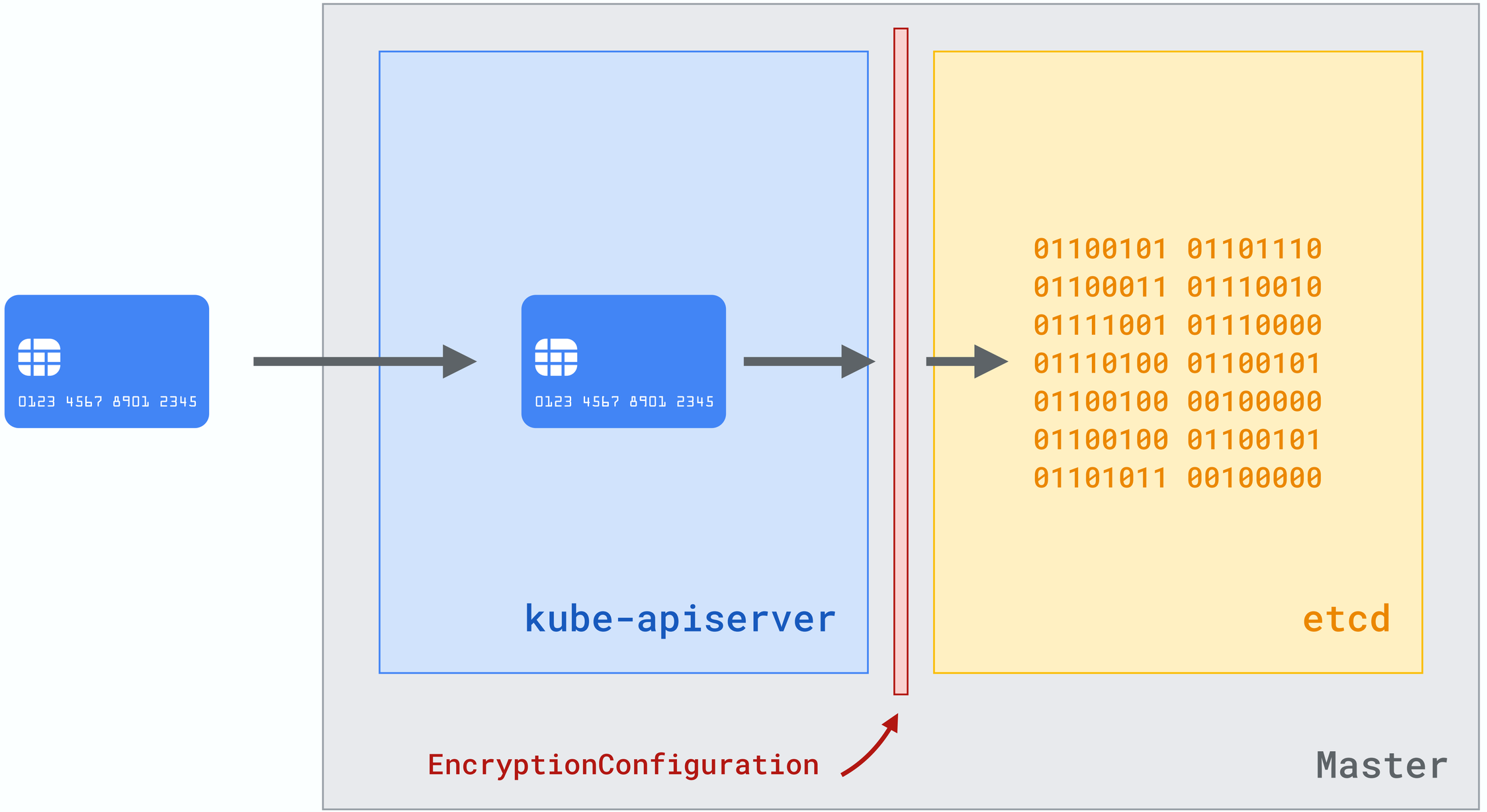
- Generate unique DEKs for each data entry
- Crypto-shred - revoke KEK and data is gone
- Easy versioning and rotation

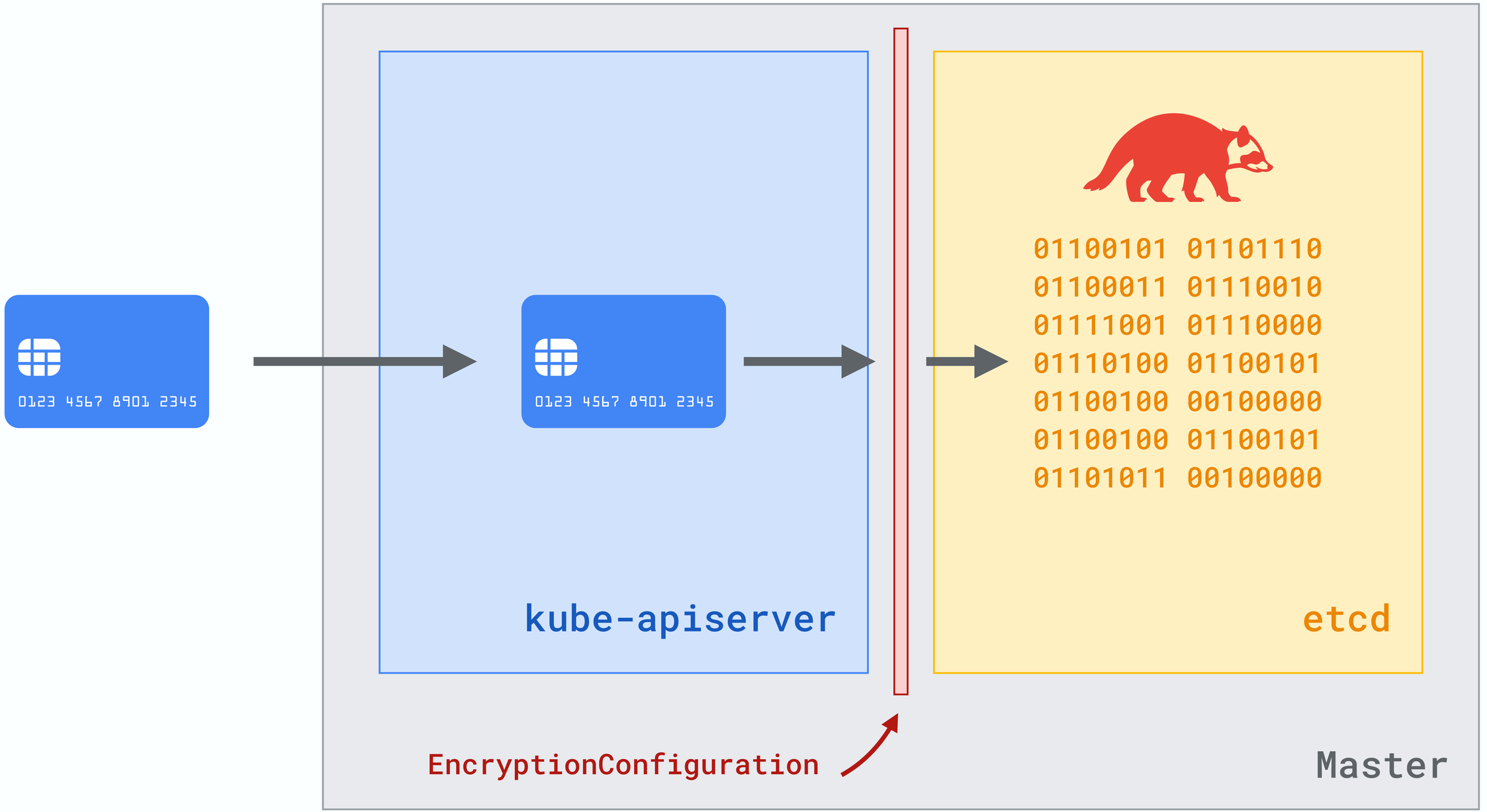
Kubernetes 1.7

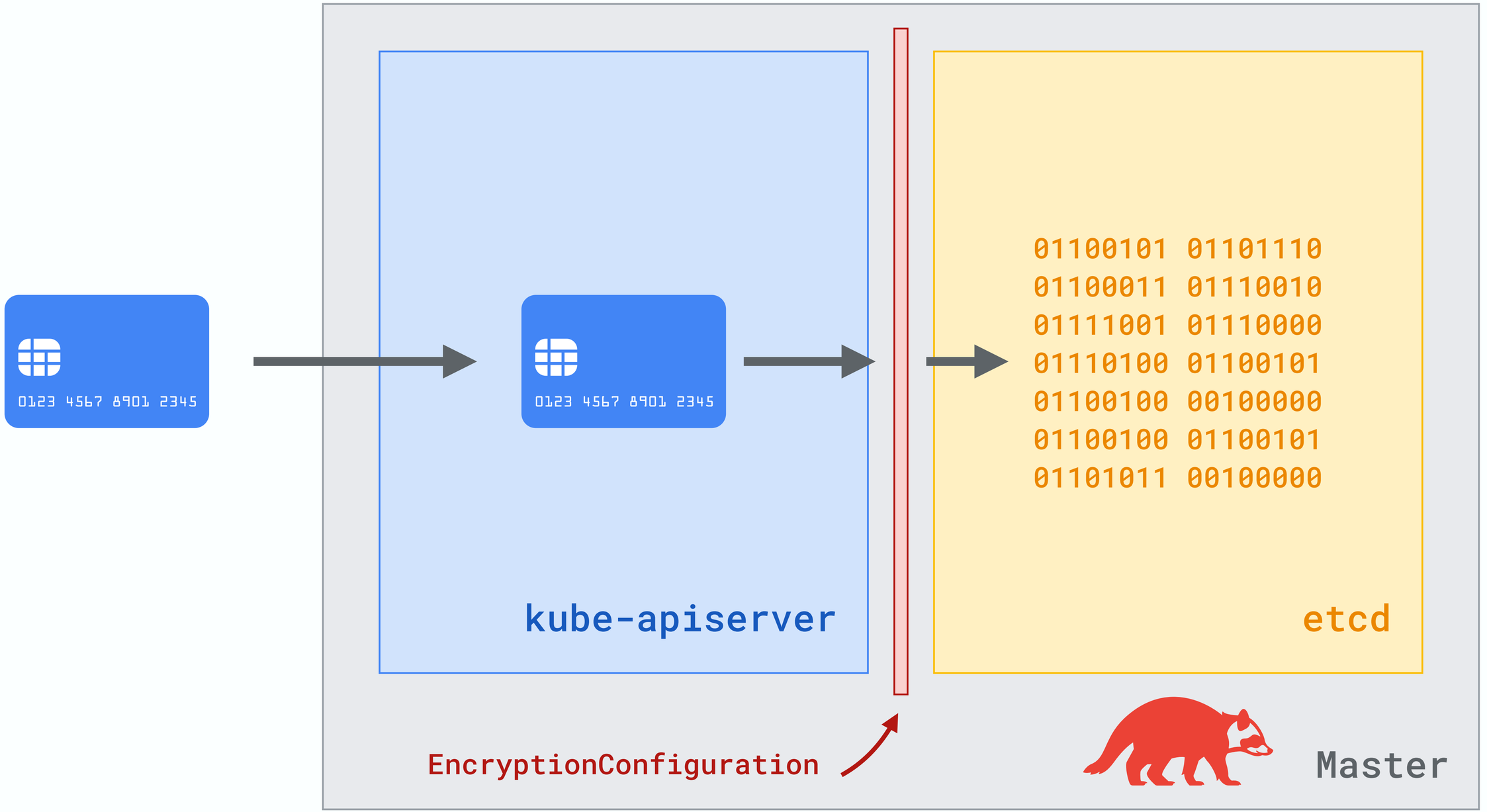
Envelope encryption

```yaml
kind: EncryptionConfiguration
apiVersion: apiserver.config.k8s.io/v1
resources:
- resources:
  - secrets
  providers:
  - aescbc:
    keys:
    - name: key1
      secret: 9RlIhvmh1e6+Ixv0CjyUkA==
    - name: key2
      secret: u+aswHTypAyoRKH5/P0r5A==
  - secretbox:
    keys:
    - name: key1
      secret: 9aHuiH/wrlmWEXZp9br4og==
```

```
./kube-apiserver \
  --encryption-provider-config=/etc/encryption-config.yaml \
  --other-options...
```


0123 4567 8901 2345
0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
kube-apiserver
etcd
EncryptionConfiguration
Master


0123 4567 8901 2345
0123 4567 8901 2345
kube-apiserver
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
etcd
EncryptionConfiguration
Master


0123 4567 8901 2345
0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
kube-apiserver
etcd
EncryptionConfiguration
Master

Drawbacks

- Need to generate keys yourself
- Key management is your responsibility
- Rotation is a manual process (and tedious)
- No HSM integration

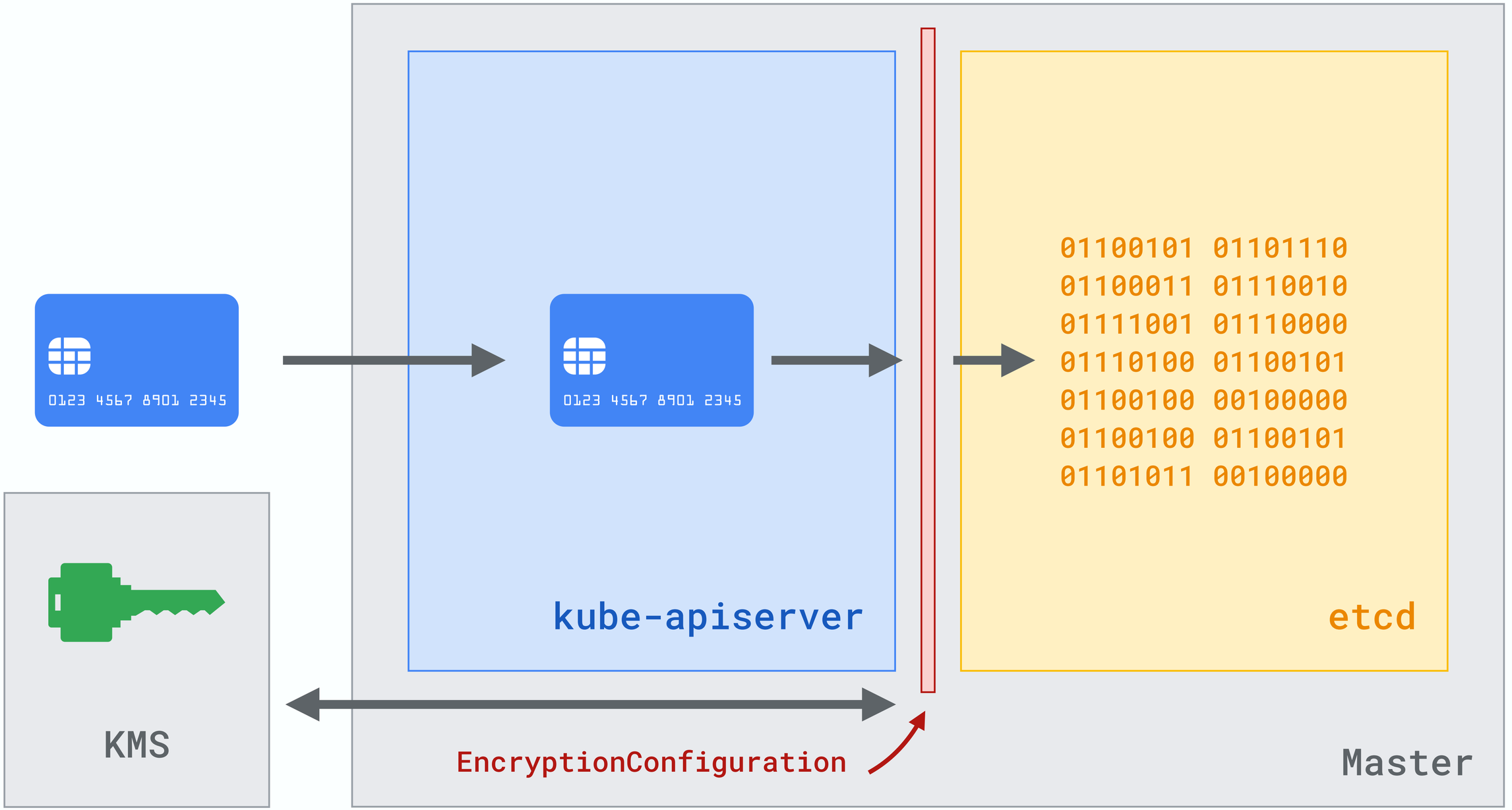
Drawbacks

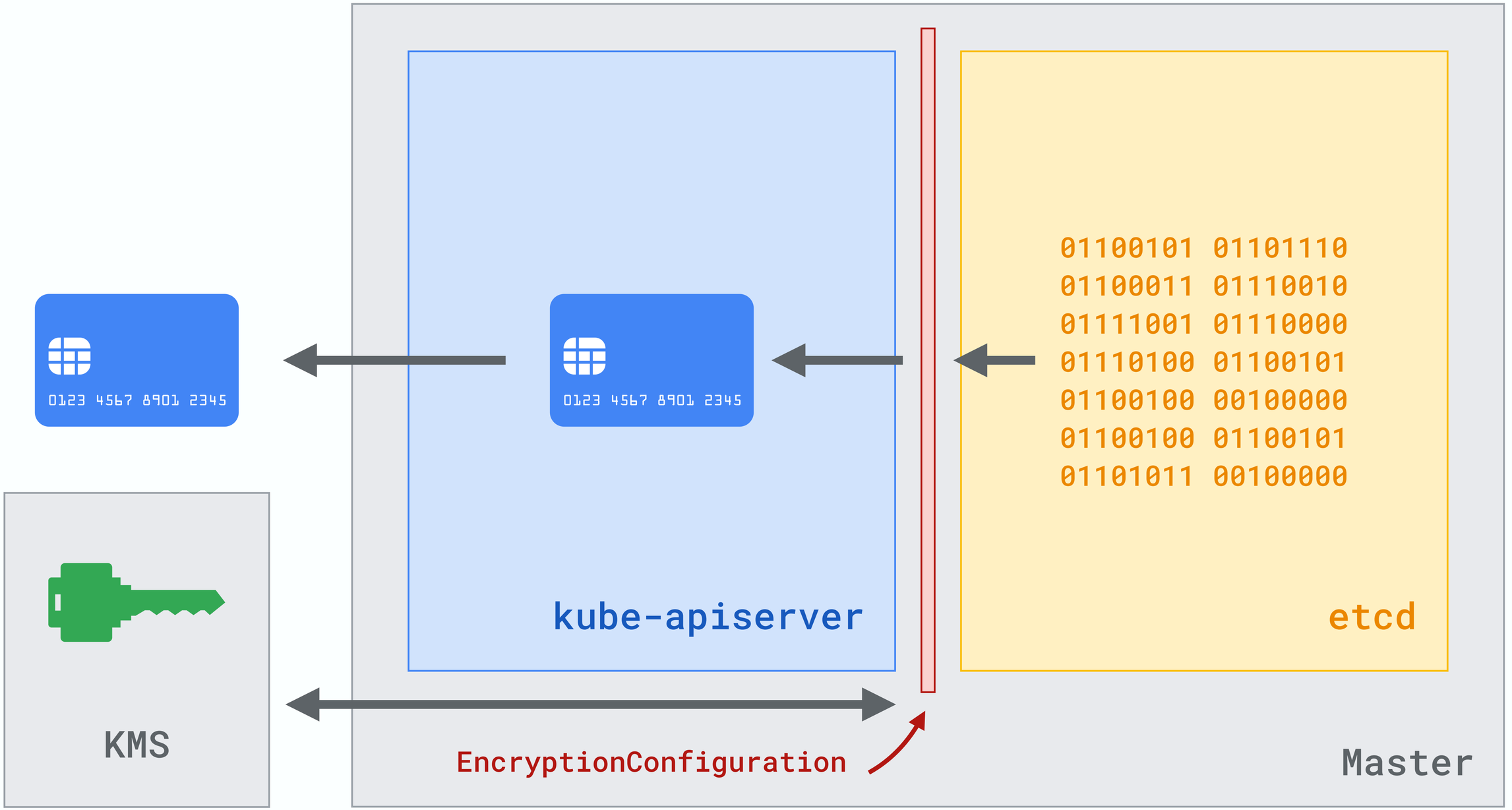
The underlying encryption keys are still stored in plaintext on the filesystem!

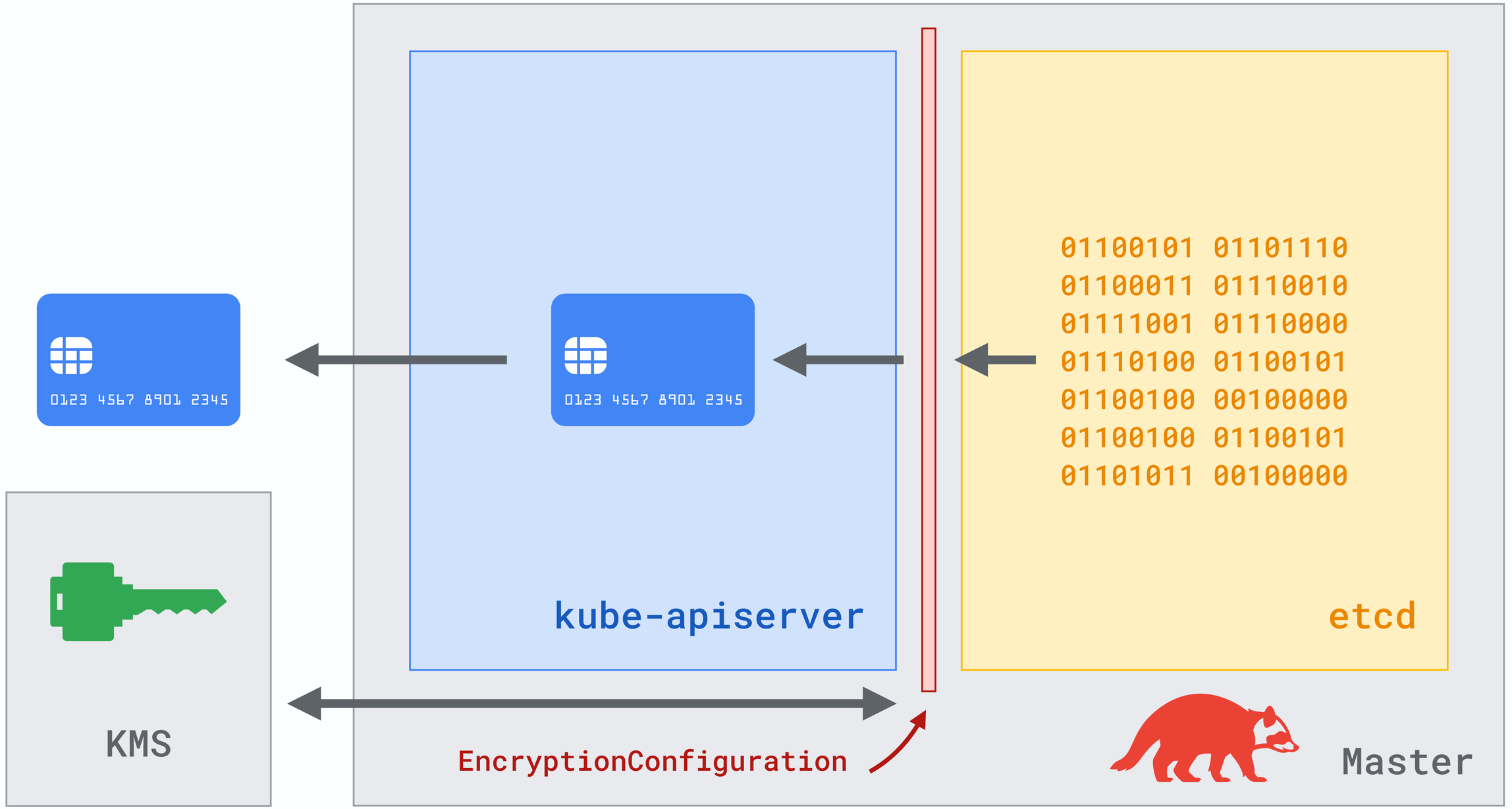
Kubernetes 1.10

KMS encryption providers

```yaml
kind: EncryptionConfiguration
apiVersion: apiserver.config.k8s.io/v1
resources:
- resources:
  - secrets
  providers:
  - kms:
      name: myKmsPlugin
      endpoint: unix:///tmp/kms-socketfile.sock
      cachesize: 100
```


0123 4567 8901 2345
0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
kube-apiserver
etcd
KMS
EncryptionConfiguration
Master


0123 4567 8901 2345
0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
kube-apiserver
etcd
KMS
EncryptionConfiguration
Master


0123 4567 8901 2345
0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
kube-apiserver
etcd
KMS
EncryptionConfiguration
Master

Existing plugins (GitHub)

- GoogleCloudPlatform/k8s-cloudkms-plugin
- Azure/kubernetes-kms
- kubernetes-sigs/aws-encryption-provider
- oracle/kubernetes-vault-kms-plugin

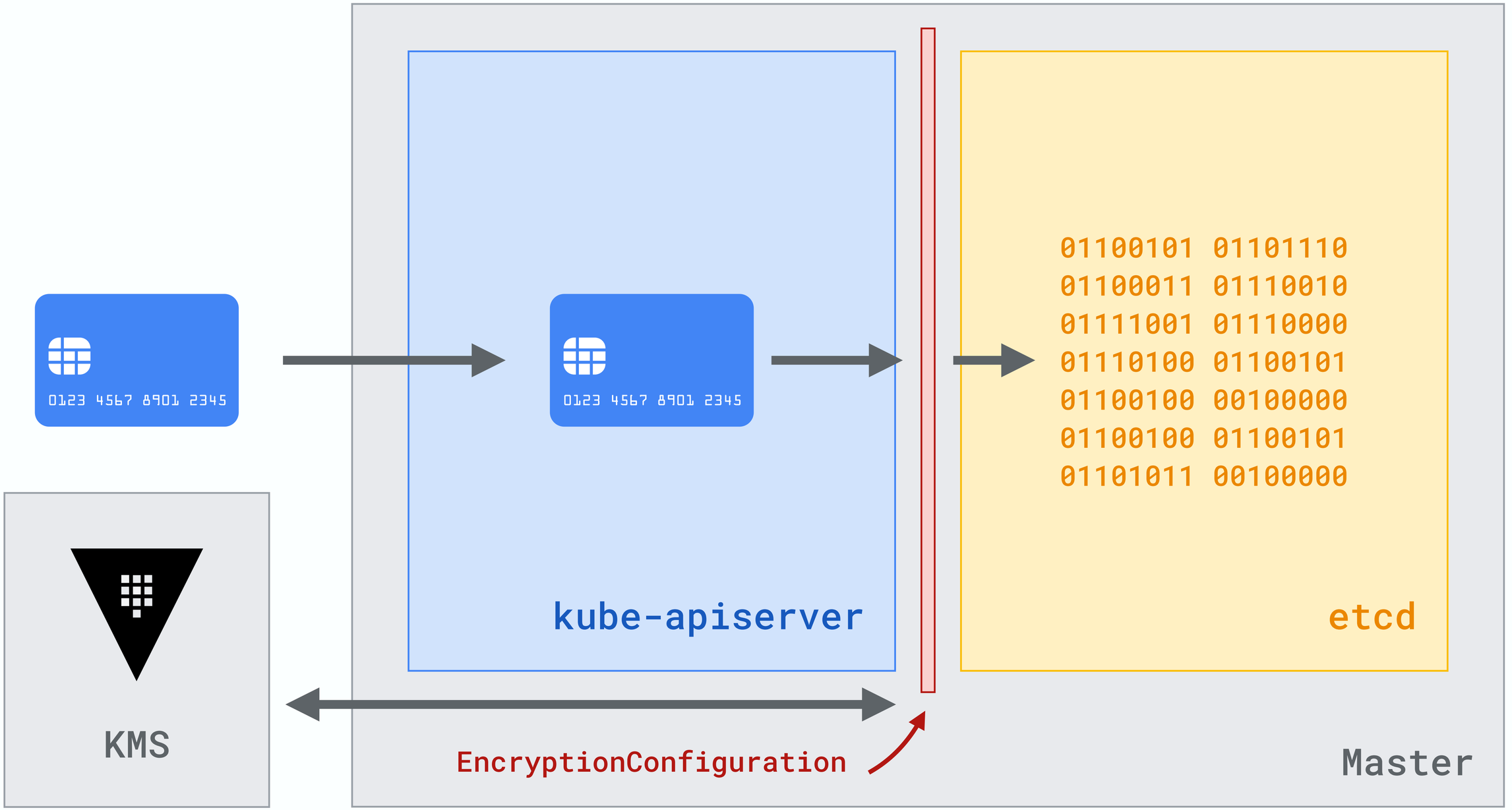
GKE Integration (beta)

```
gcloud beta container clusters create my-cluster
  --database-encryption-key-location us-east1
  --database-encryption-key-keyring my-keyring
  --database-encryption-key my-crypto-key
```

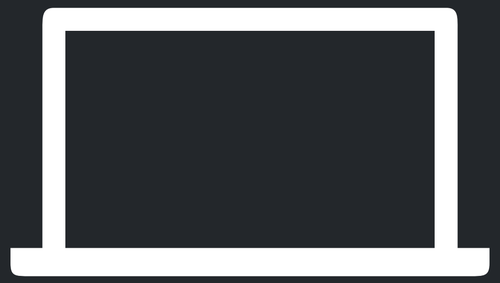
Initial secret problem?

- IAM can solve the "first secret" problem
- Delegate PAM to the cloud provider via IAM
- Separate concerns: etcd nodes don't need IAM permissions to talk to KMS

Vault


0123 4567 8901 2345
0123 4567 8901 2345
01100101 01101110
01100011 01110010
01111001 01110000
01110100 01100101
01100100 00100000
01100100 01100101
01101011 00100000
kube-apiserver
etcd
KMS
EncryptionConfiguration
Master

Demo



Summary

Summary

- Use at least two layers of encryption
- Rotate keys regularly
- Leverage envelope encryption
- Protect K8S secrets using an external KMS

Thanks!

@sethvargo

Developer Relations Engineer