Extending Kubernetes with Storage Transformers

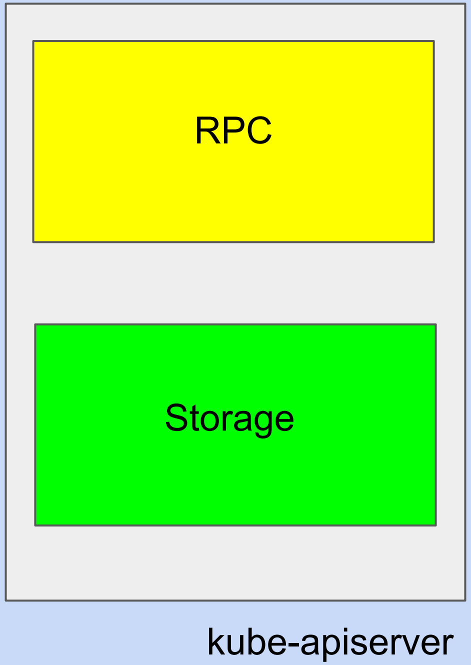
Andrew Lytvynov awly@google.com

KubeCon China 2019

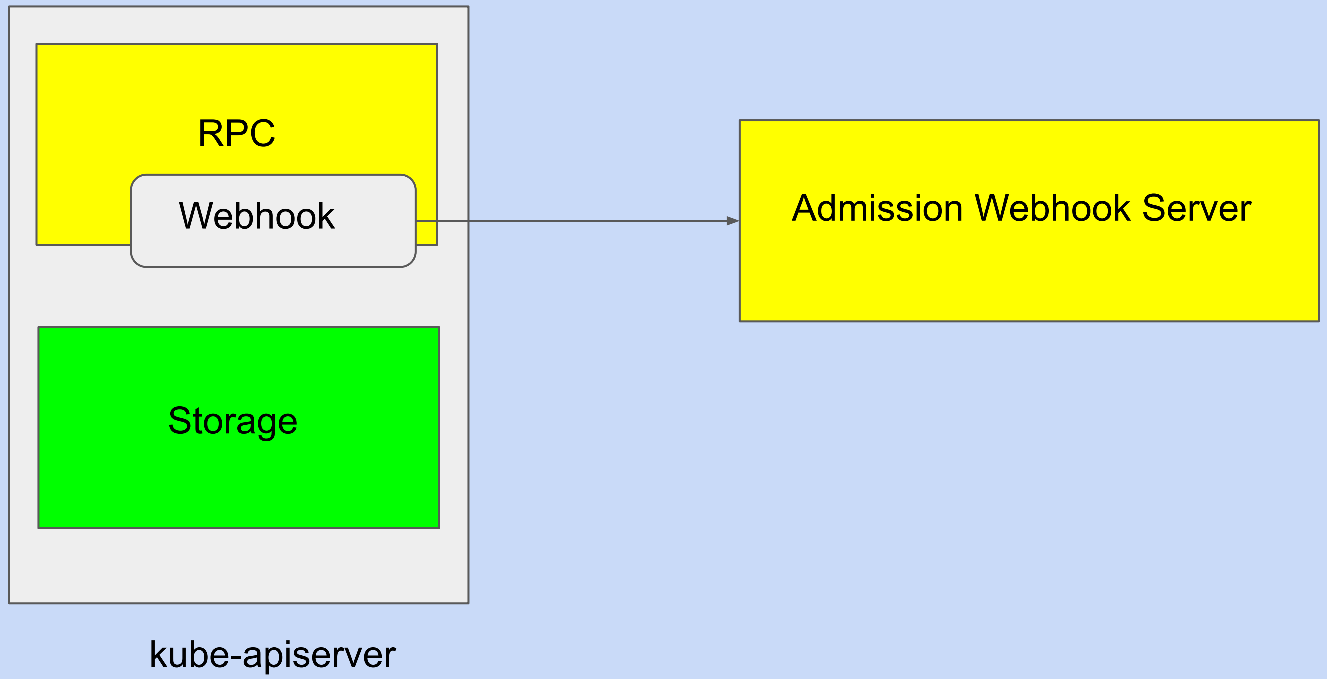
Agenda

1. What are Transformers
2. Why do we need them
3. How do you implement them
4. How do we secure them

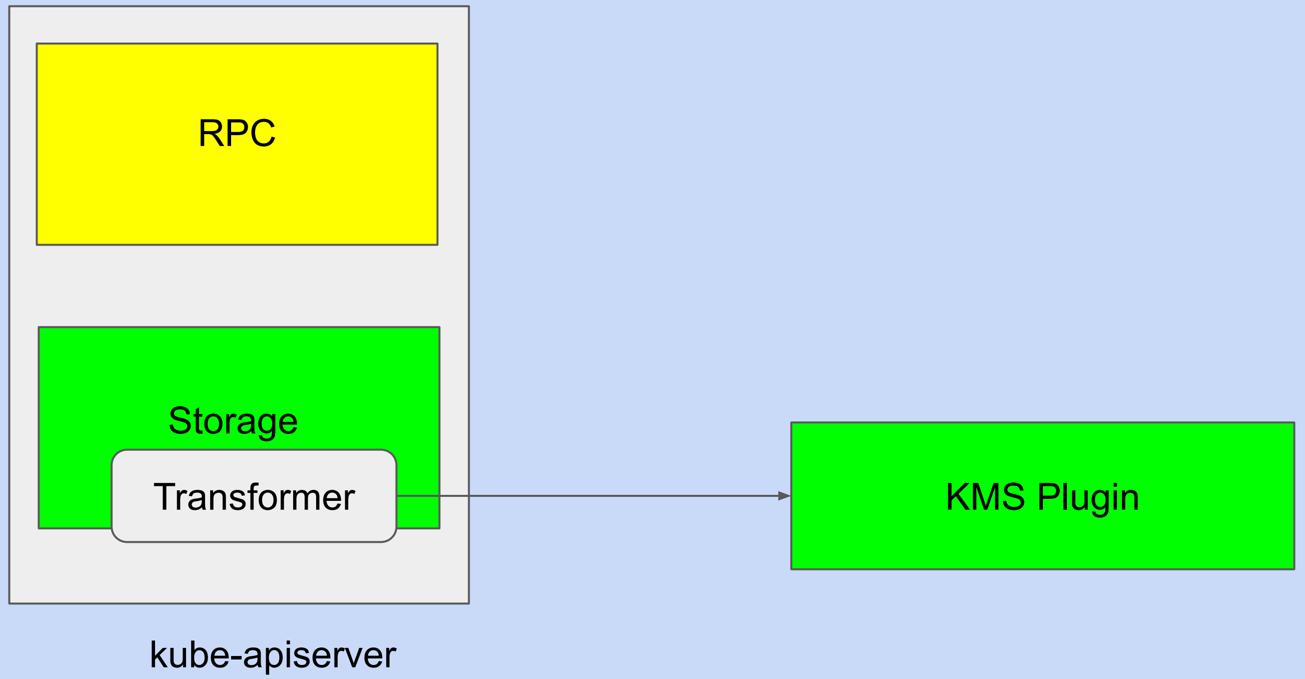
RPC vs. Storage layers of kube-apiserver



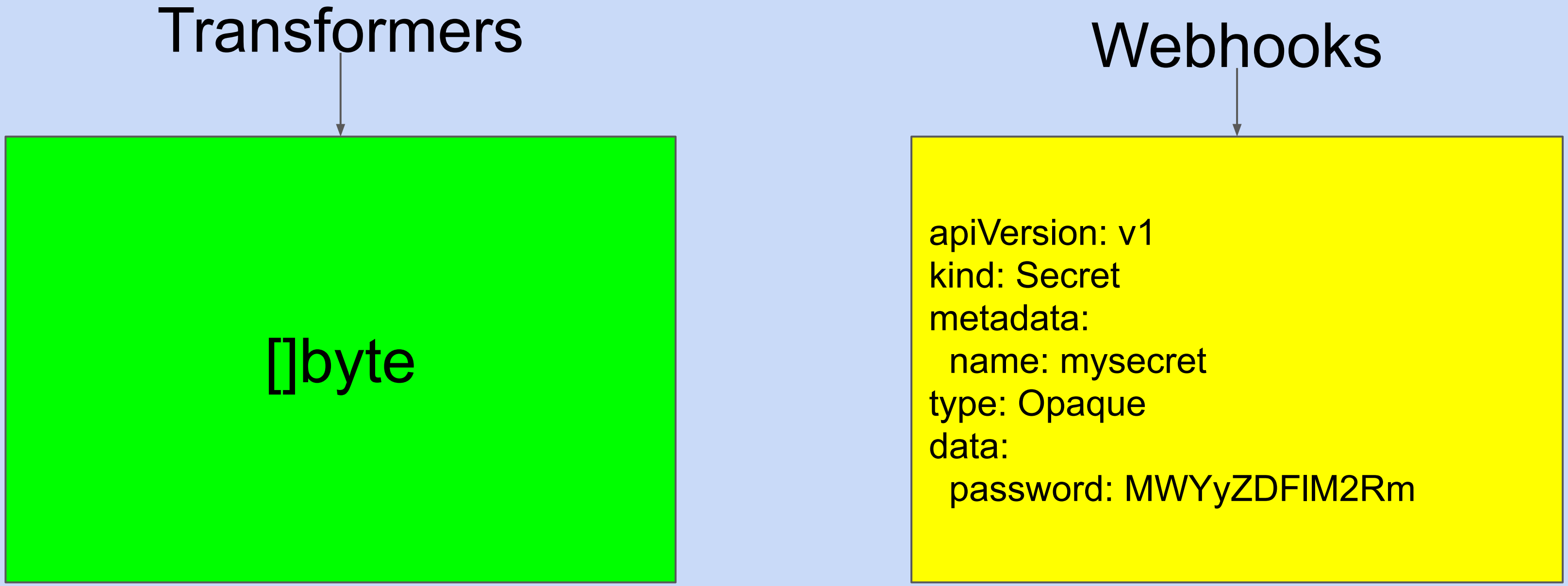
Extensibility at the RPC layer



Extensibility at the Storage layer

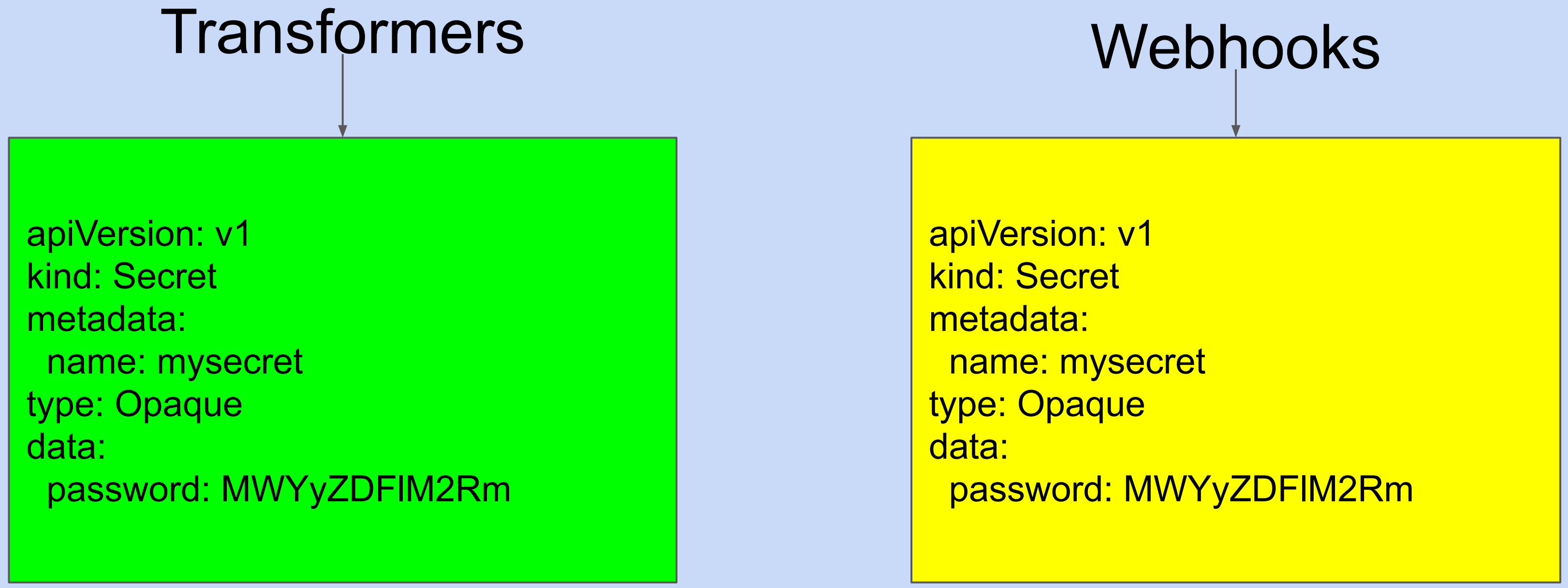


Schema vs []byte



```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: MWYyZDFlM2Rm
```

Convention



```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: MWYyZDFlM2Rm
```

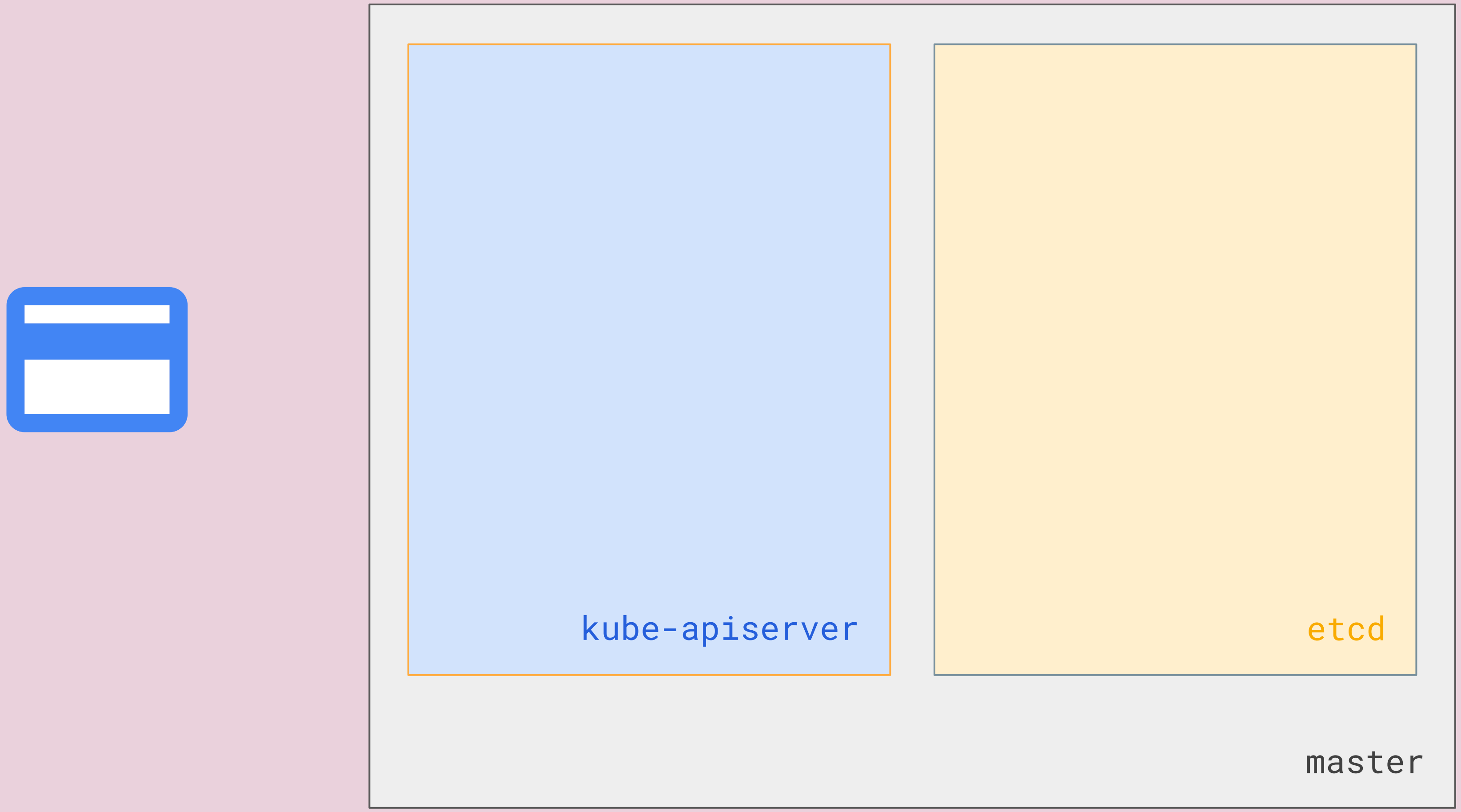
```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: MWYyZDFlM2Rm
```

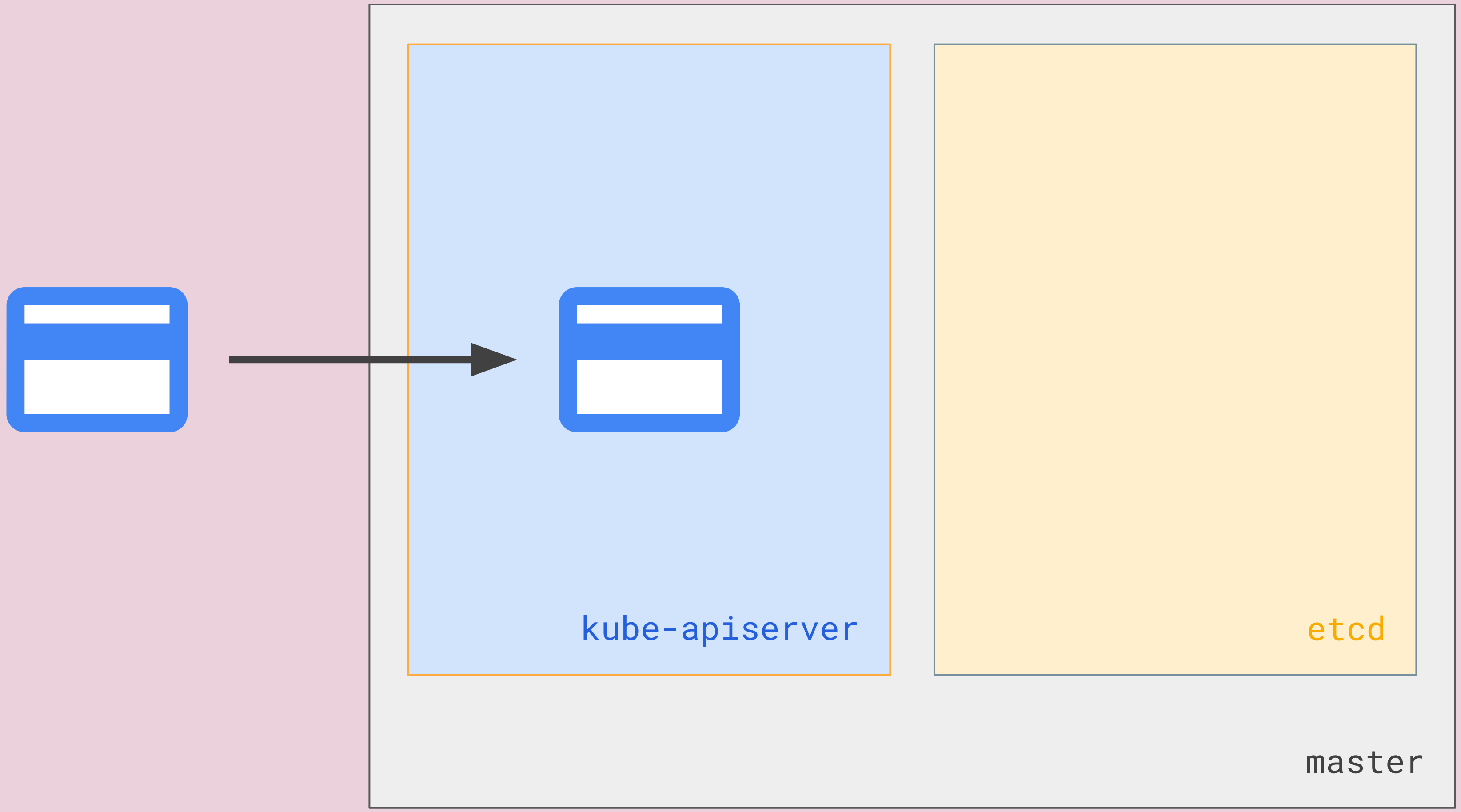
Why a Kubecon talk about Transformers

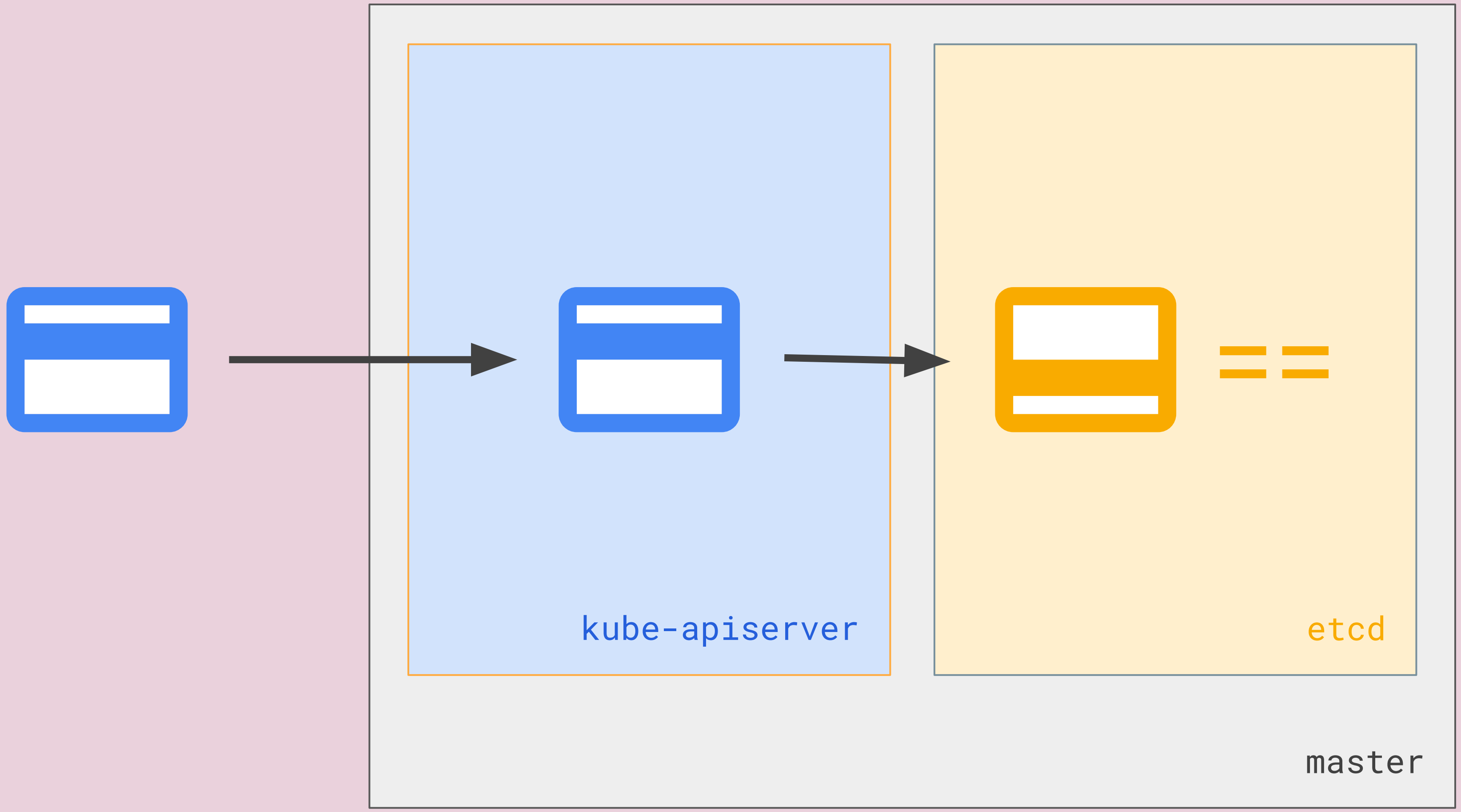
- Explain the feature
- Increase contribution
- Share lessons learned
- Spark new ideas

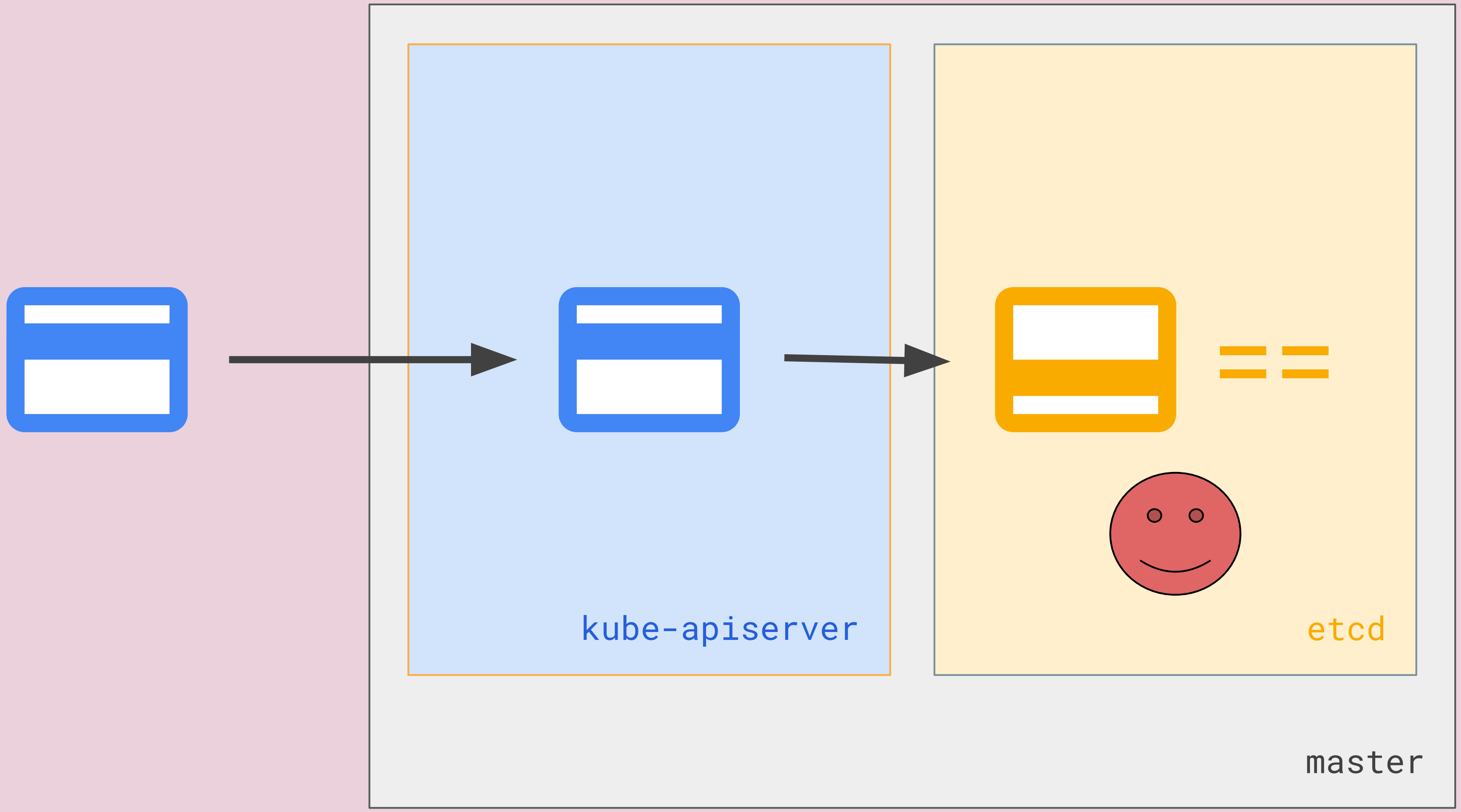
Motivating Problem - Encrypting Secrets at Rest

A default OSS Kubernetes setup is not encrypted by default.
Secrets are stored in plaintext.

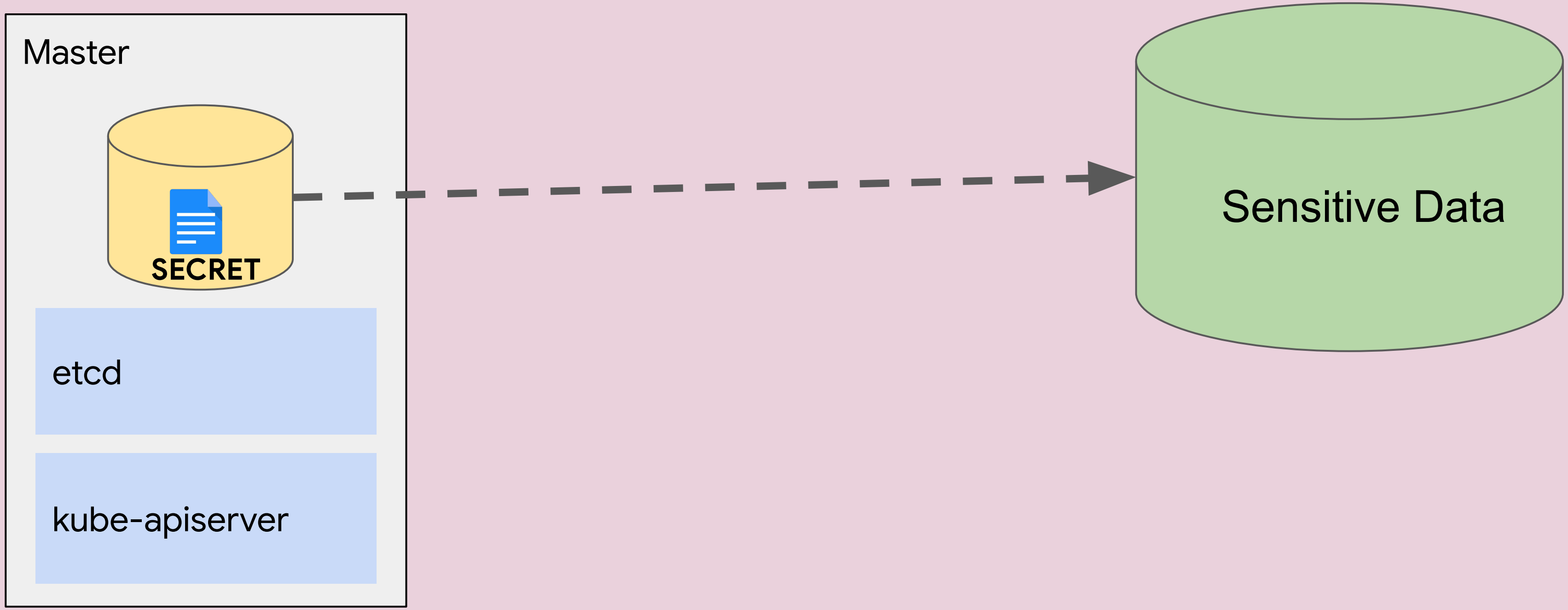

kube-apiserver
etcd
master


kube-apiserver
etcd
master

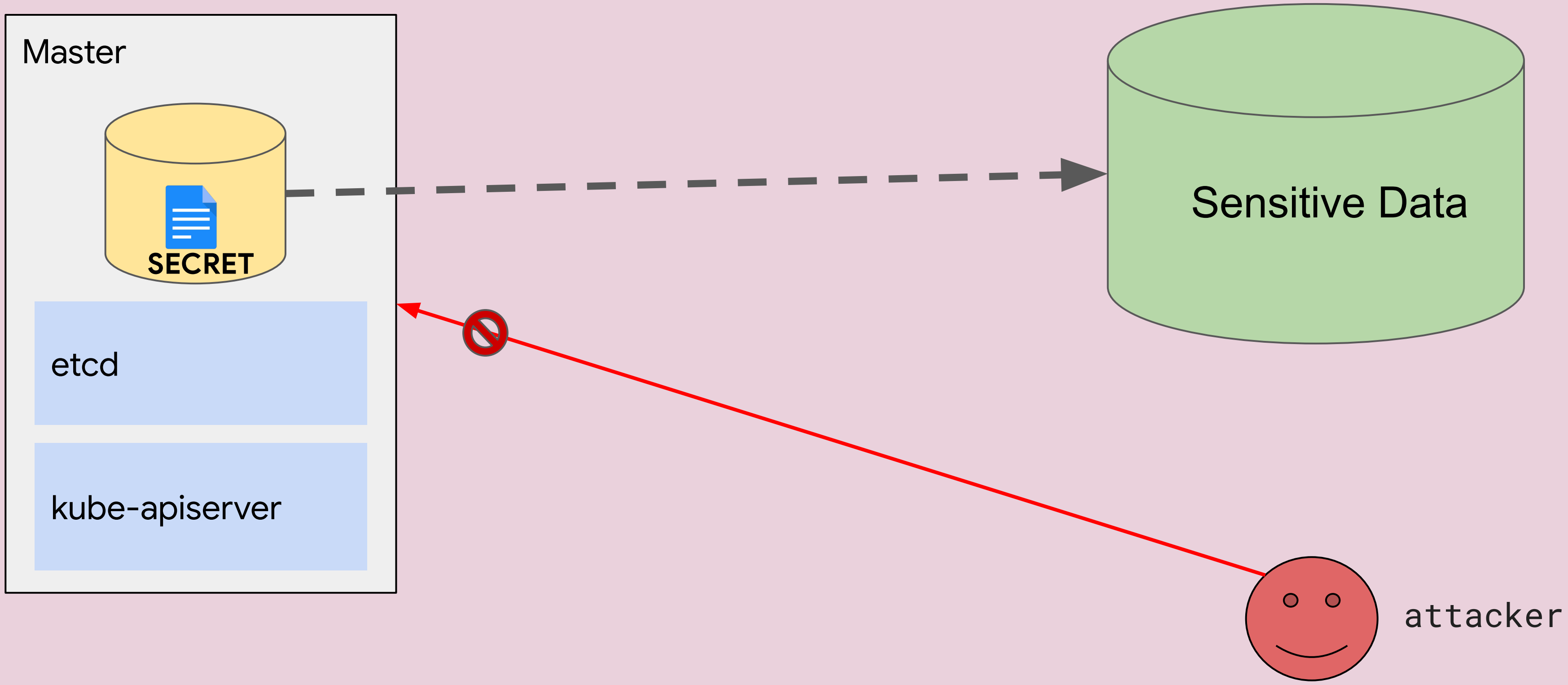

kube-apiserver
etcd
master


kube-apiserver
etcd
master

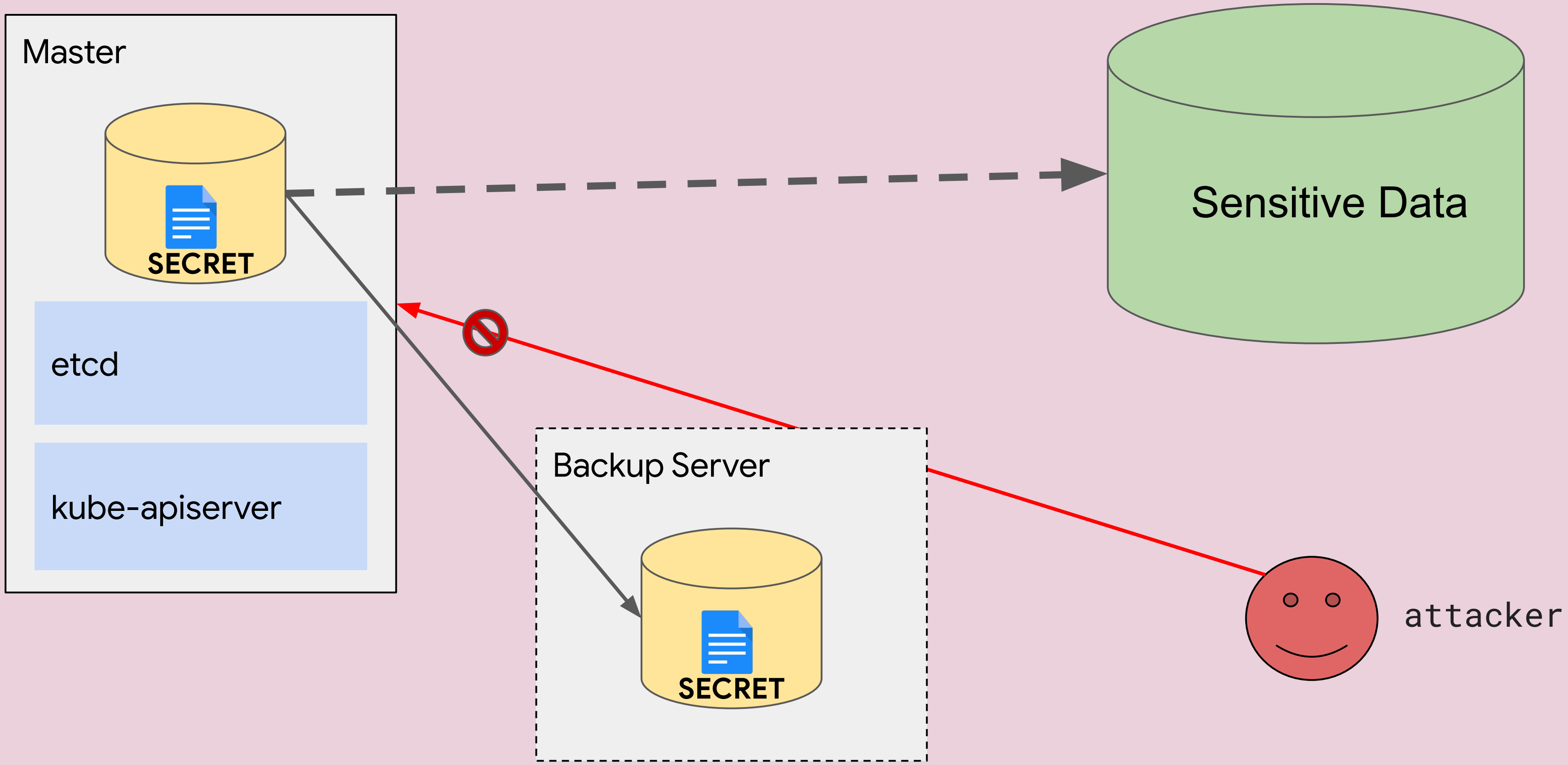
Offline attacks



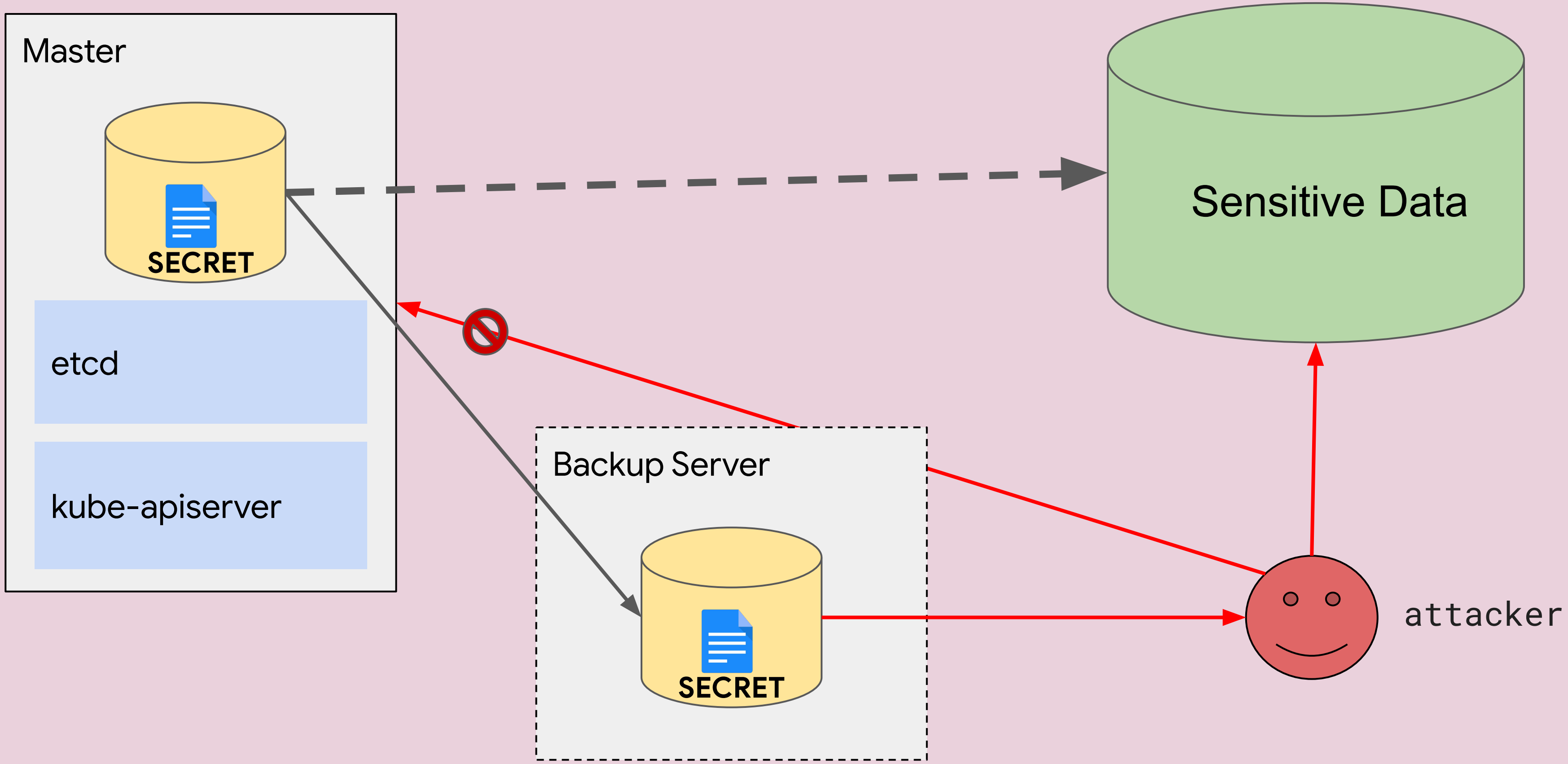
Offline attacks



Offline attacks



Offline attacks



Demo: fancy tools?

Implementing Storage Transformers

https://github.com/awly/kubernetes/commits/kubecon-china-transformers

Step #1: Implement Transformer Interface

k8s.io/apiserver/pkg/storage/value/encrypt/mytransformer/

```go
type Transformer interface {
    TransformFromStorage(data []byte, context Context) (out []byte, stale bool, err error)
    TransformToStorage(data []byte, context Context) (out []byte, err error)
}
```

https://github.com/awly/kubernetes/commit/214b616b152aa22a851b37923a7314de834dc104

Step #2: Create your YAML config structure

```yaml
kind: EncryptionConfiguration
apiVersion: apiserver.config.k8s.io/v1
resources:
- resources:
  - secrets
  providers:
  - myProvider:
      key: key1
      field2: value2
```

https://github.com/awly/kubernetes/commit/d1c32f5766931f08567dda318b0b30eb8390fd60

Step #2: Create your YAML config structure

k8s.io/apiserver/pkg/apis/config/types.go
k8s.io/apiserver/pkg/apis/config/v1/types.go

```
type MyConfiguration struct {
    Key Key `json:"key"`
    Field2 Type2 `json:"field2"`
    ...
}
```

https://github.com/awly/kubernetes/commit/d1c32f5766931f08567dda318b0b30eb8390fd60

Step #3: Add your type to `ProviderConfiguration`

k8s.io/apiserver/pkg/apis/config/types.go
k8s.io/apiserver/pkg/apis/config/v1/types.go

```
type ProviderConfiguration struct {
    AESGCM *AESConfiguration
    AESCBC *AESConfiguration
    Secretbox *SecretboxConfiguration
    Identity *IdentityConfiguration
    KMS *KMSConfiguration
    MyProvider *MyConfiguration
}
```

https://github.com/awly/kubernetes/commit/39397c8bb87e4b17904d64820355ebe9bb6e6c45

Prefix Transformer

```
$ cat ${ETCD_DATA} | grep -A 2 -a db-password
...
 {2e+1^f)'=[lXr;%v'}efault/db-password33
*k8s:enc:my:v1:key1:v;c[yb;5;;PzV|&v̏!x@\w5;QQ&eXTHrQ□V@
J`(4_
...
```

Prefix Transformer

```
type PrefixTransformer struct {
        Prefix      []byte
        Transformer Transformer
}
```

Step #4: Define your prefix

k8s.io/apiserver/pkg/server/options/encryptionconfig/config.go

```
const (
    aesCBCTransformerPrefixV1    = "k8s:enc:aescbc:v1:"
    aesGCMTransformerPrefixV1    = "k8s:enc:aesgcm:v1:"
    secretboxTransformerPrefixV1 = "k8s:enc:secretbox:v1:"
    kmsTransformerPrefixV1       = "k8s:enc:kms:v1:"
    myTransformerPrefixV1        = "k8s:enc:my:v1:"
)
```

https://github.com/awly/kubernetes/commit/cf128fc92e5e66582bc02c6992df065b91e6c582

Step #5: Add Init logic for your transformer

k8s.io/apiserver/pkg/server/options/encryptionconfig/config.go

```
func GetMyPrefixTransformer(config *apiserverconfig.MyConfiguration, prefix string)
  (value.PrefixTransformer, error) {

    // 1. Validate and parse fields of config.
    // 2. Create an instance of MyTransformer.

    return value.PrefixTransformer{
        Transformer: myTransformer,
        Prefix:      []byte(prefix),
    }, nil
}
```

https://github.com/awly/kubernetes/commit/a82c659f7f13450b16bf3f47c17f6566495c43c5

Demo: SM4 transformer

It works, but...

1. Key rotation is manual and requires kube-apiserver restart
2. **Key is in plaintext on disk**

Envelope Transformers

Envelope encryption

Data

Data encryption key
DEK

Key encryption key
KEK

Envelope encryption

Data

$\{\text{SECRET}\}_{\text{DEK}}$

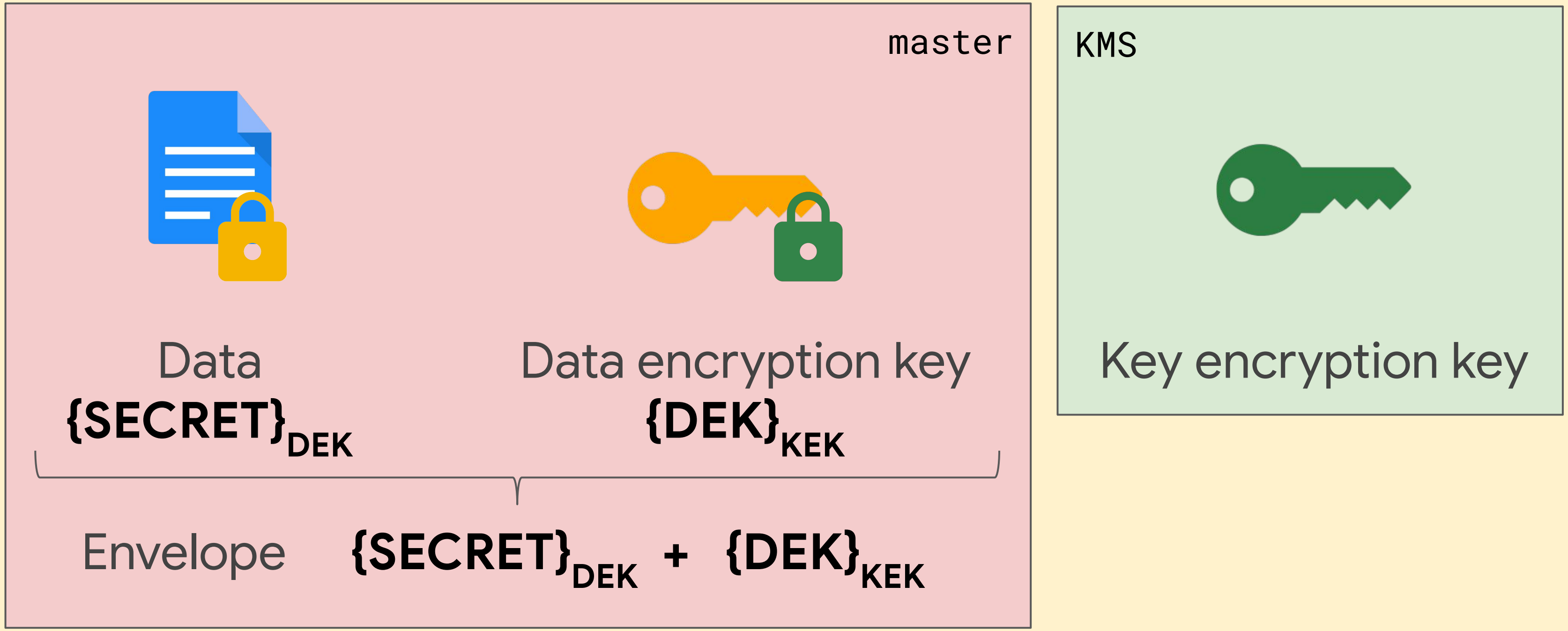
Data encryption key

$\{\text{DEK}\}_{\text{KEK}}$

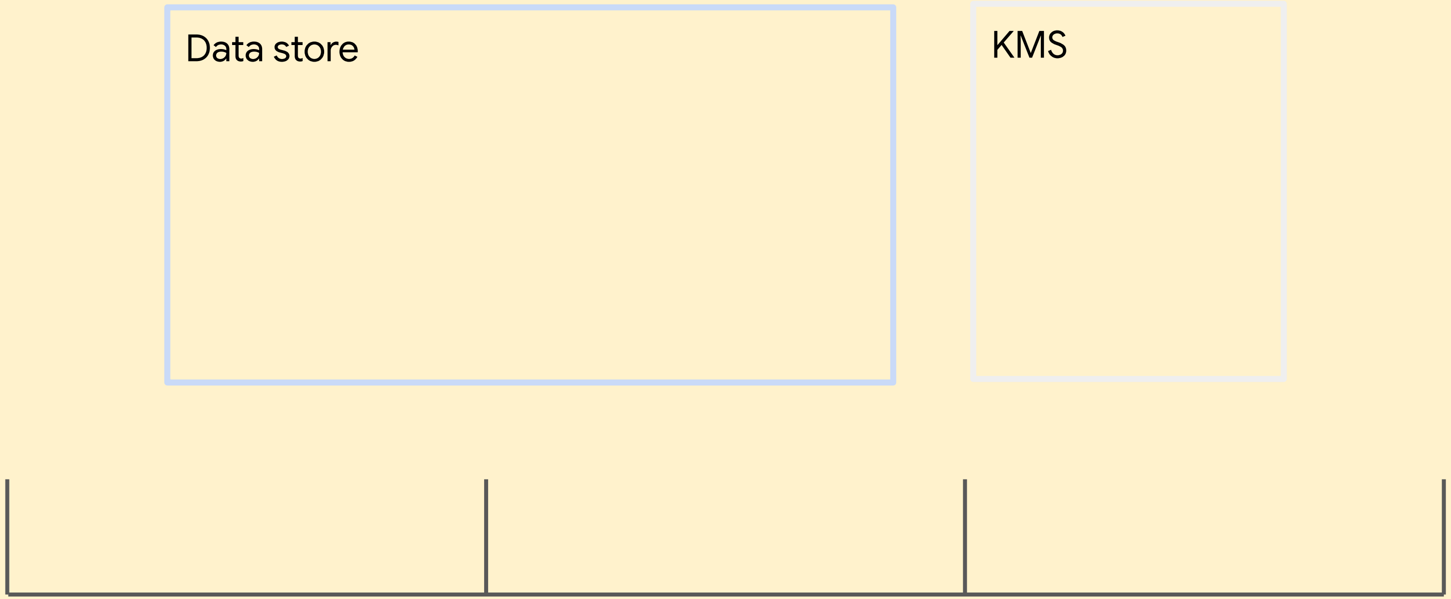
Key encryption key

Envelope $\{\text{SECRET}\}_{\text{DEK}} + \{\text{DEK}\}_{\text{KEK}}$

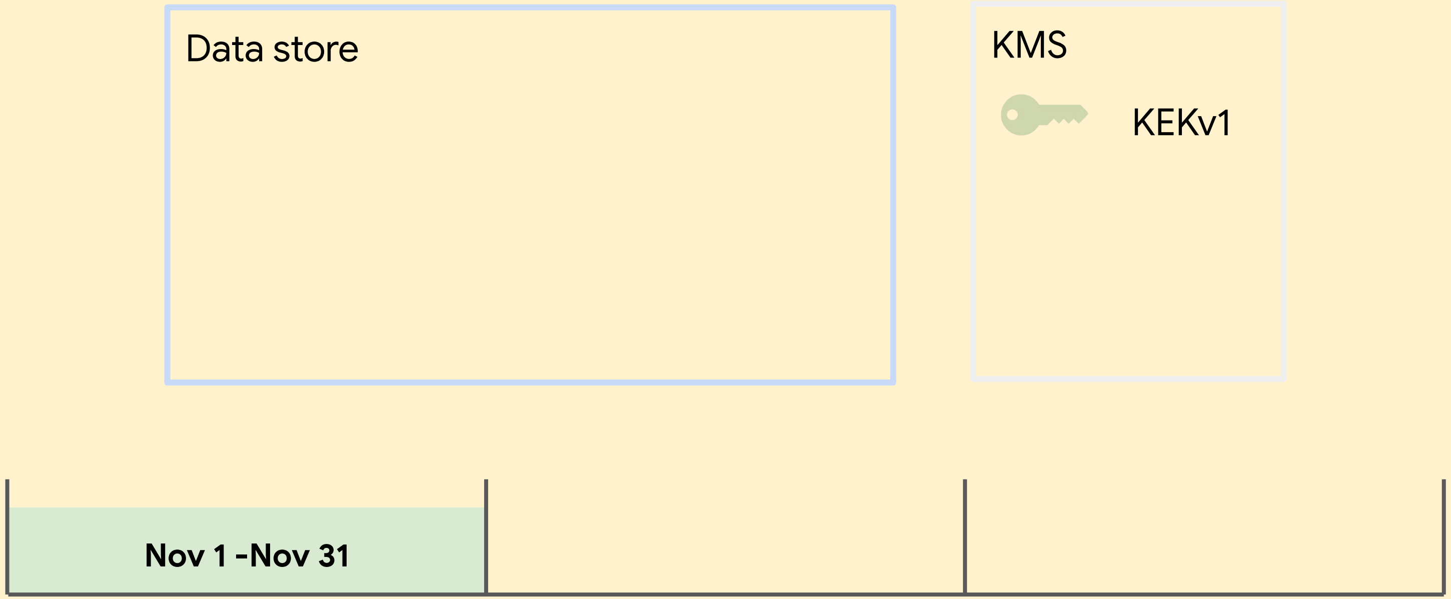
Envelope encryption



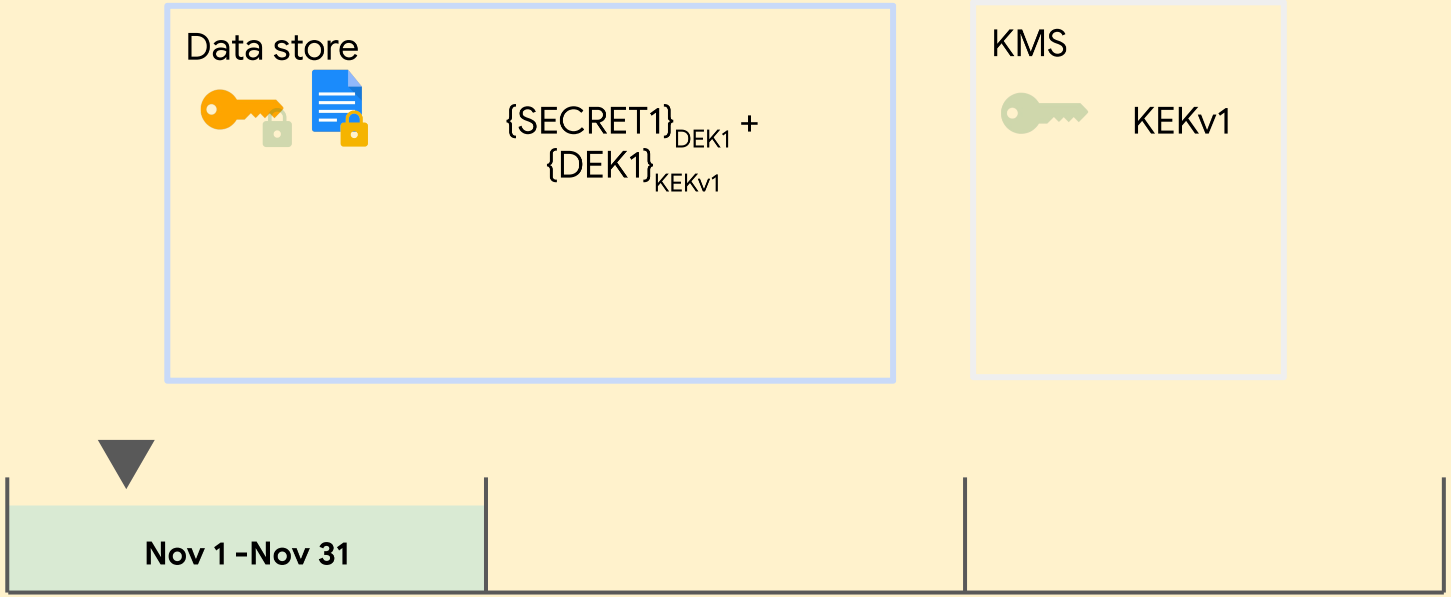
Version management



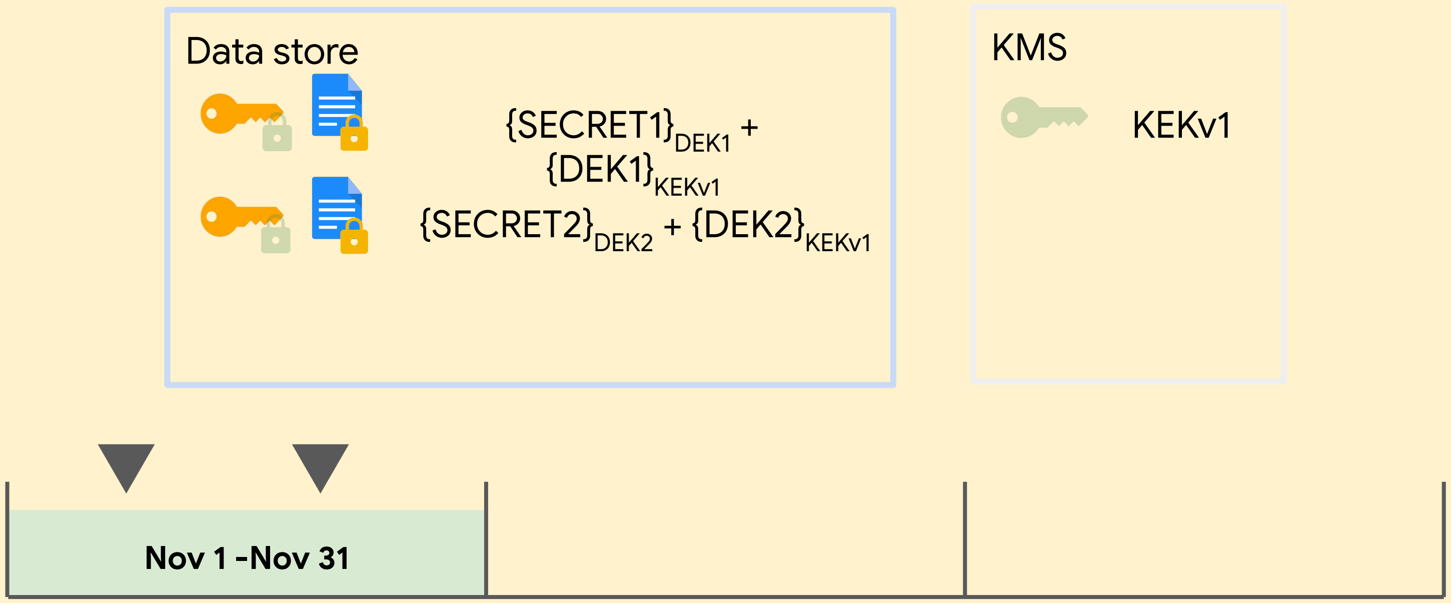
Version management



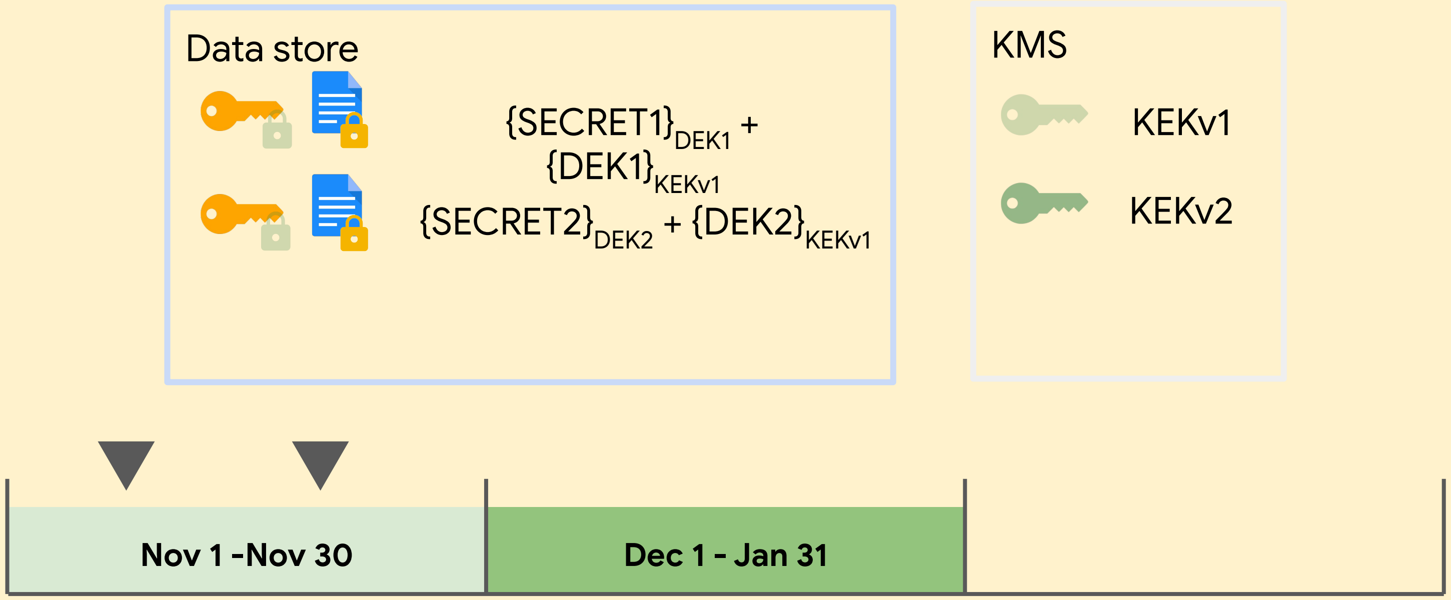
Version management



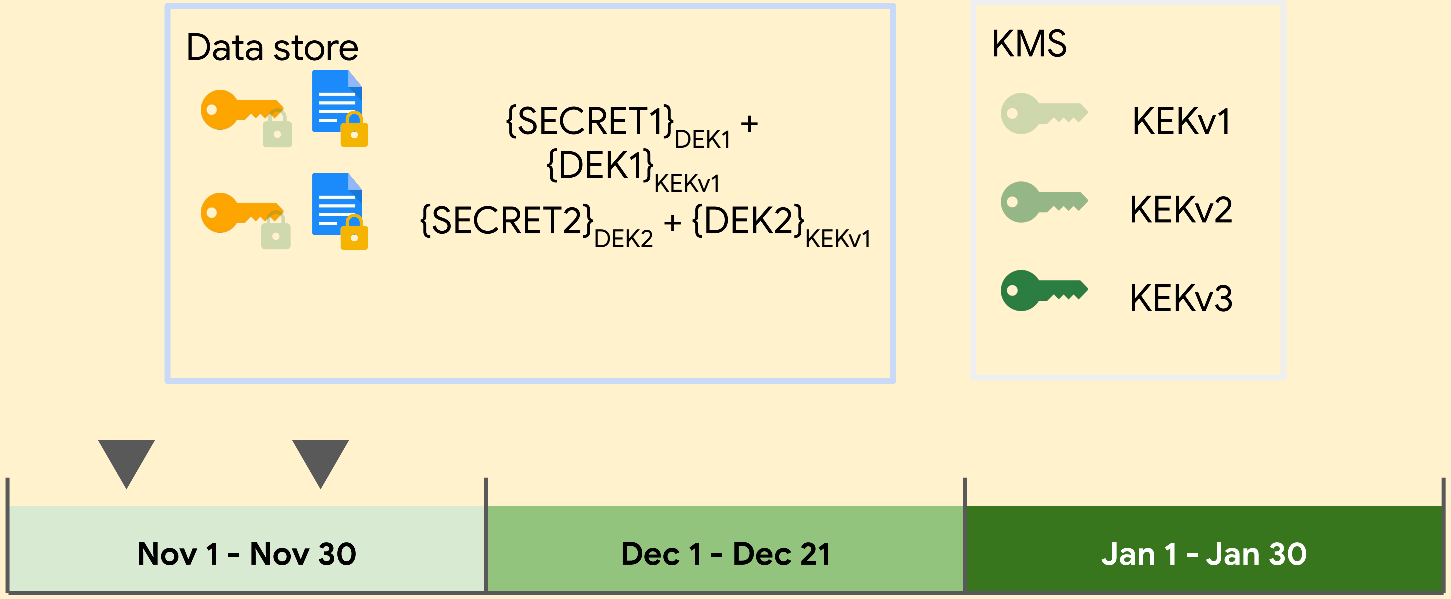
Version management


Data store
$\{SECRET1\}_{DEK1} + \{DEK1\}_{KEKv1}$
$\{SECRET2\}_{DEK2} + \{DEK2\}_{KEKv1}$
KMS
KEKv1
Nov 1 -Nov 31

Version management

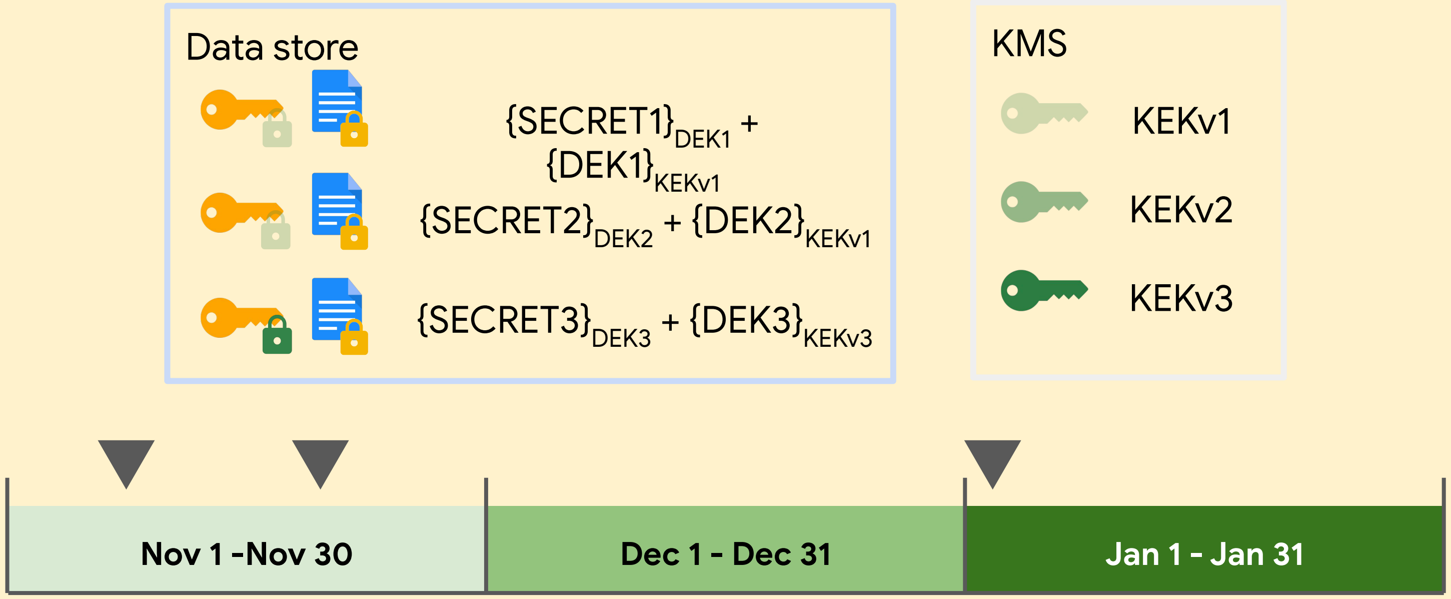

Data store
$\{SECRET1\}_{DEK1}$ + $\{DEK1\}_{KEKv1}$
$\{SECRET2\}_{DEK2}$ + $\{DEK2\}_{KEKv1}$
KMS
KEKv1
KEKv2
Nov 1 -Nov 30
Dec 1 - Jan 31

Version management

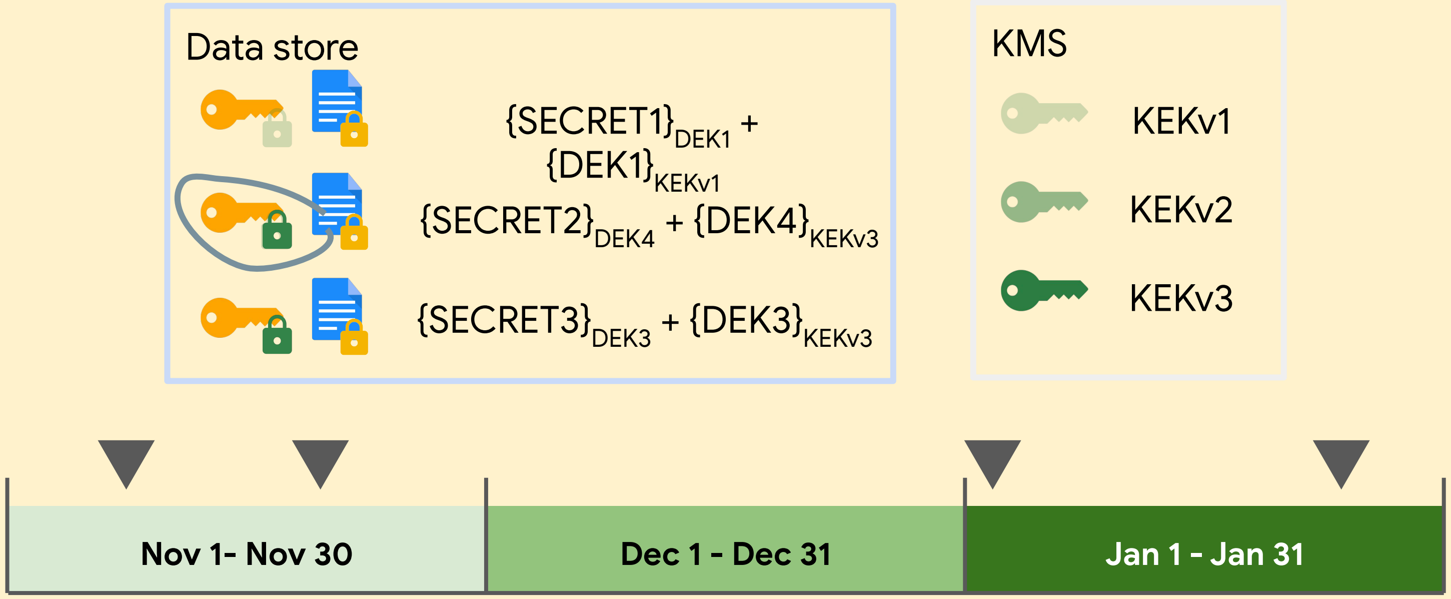


Data store
$\{SECRET1\}_{DEK1}$ + $\{DEK1\}_{KEKv1}$
$\{SECRET2\}_{DEK2}$ + $\{DEK2\}_{KEKv1}$
KMS
KEKv1
KEKv2
KEKv3
Nov 1 - Nov 30
Dec 1 - Dec 21
Jan 1 - Jan 30

Version management


Data store
$\{SECRET1\}_{DEK1}$ + $\{DEK1\}_{KEKv1}$
$\{SECRET2\}_{DEK2}$ + $\{DEK2\}_{KEKv1}$
$\{SECRET3\}_{DEK3}$ + $\{DEK3\}_{KEKv3}$
KMS
KEKv1
KEKv2
KEKv3
Nov 1 -Nov 30
Dec 1 - Dec 31
Jan 1 - Jan 31

Version management


Data store
$\{SECRET1\}_{DEK1}$ + $\{DEK1\}_{KEKv1}$
$\{SECRET2\}_{DEK4}$ + $\{DEK4\}_{KEKv3}$
$\{SECRET3\}_{DEK3}$ + $\{DEK3\}_{KEKv3}$
KMS
KEKv1
KEKv2
KEKv3
Nov 1- Nov 30
Dec 1 - Dec 31
Jan 1 - Jan 31

Implementing Envelope Transformer

Re-using Envelope Transformer

k8s.io/apiserver/pkg/storage/value/encrypt/envelope/envelope.go

```go
func NewEnvelopeTransformer(
     envelopeService Service,
     cacheSize int,
     baseTransformerFunc func(cipher.Block) value.Transformer,
) (value.Transformer, error)
```

Re-using Envelope Transformer

k8s.io/apiserver/pkg/storage/value/encrypt/envelope/envelope.go

```
type Service interface {
    Decrypt(data []byte) ([]byte, error)
    Encrypt(data []byte) ([]byte, error)
}
```

KMS Plugins

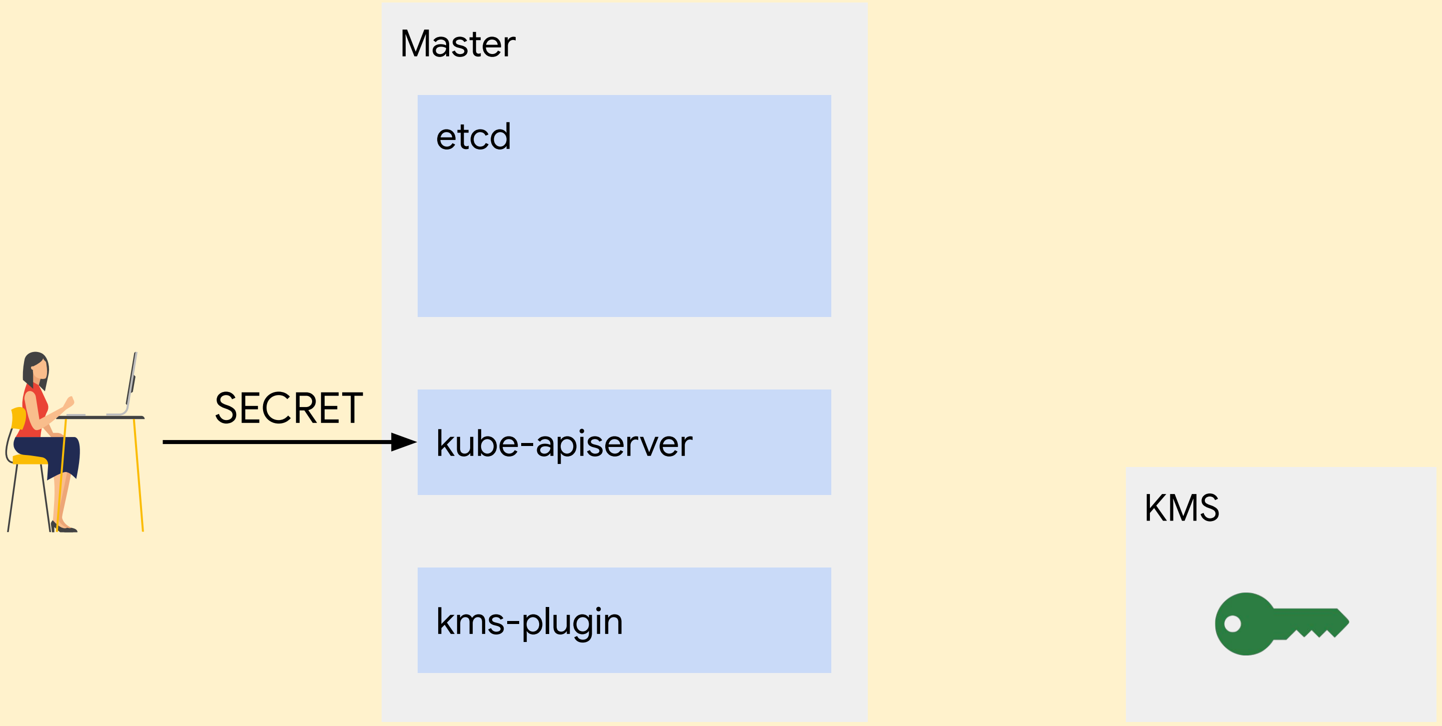
KMS encryption configuration

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

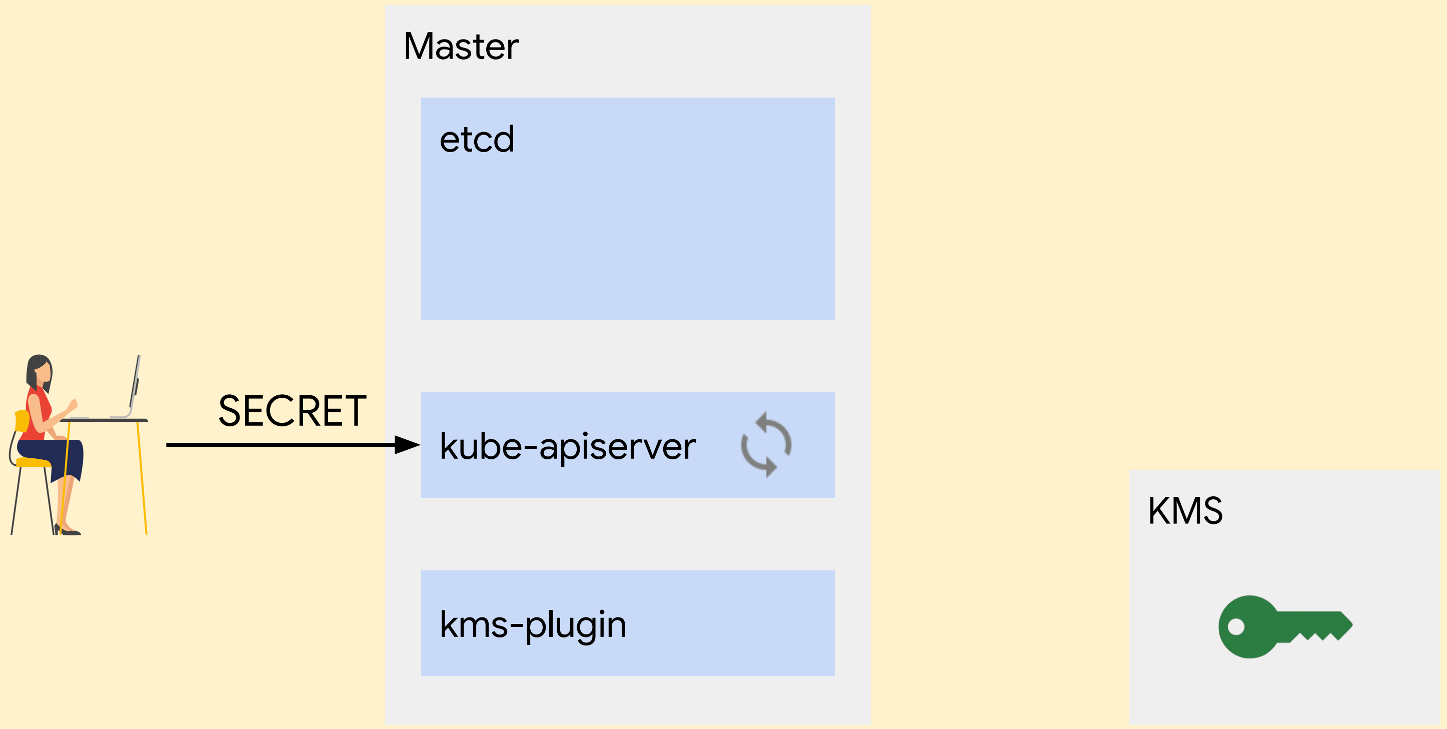
gRPC Service

```
service KeyManagementService {
   // Version returns the runtime name and runtime version of the KMS provider.
   rpc Version(VersionRequest) returns (VersionResponse) {}
   // Execute decryption operation in KMS provider.
   rpc Decrypt(DecryptRequest) returns (DecryptResponse) {}
   // Execute encryption operation in KMS provider.
   rpc Encrypt(EncryptRequest) returns (EncryptResponse) {}
}
```

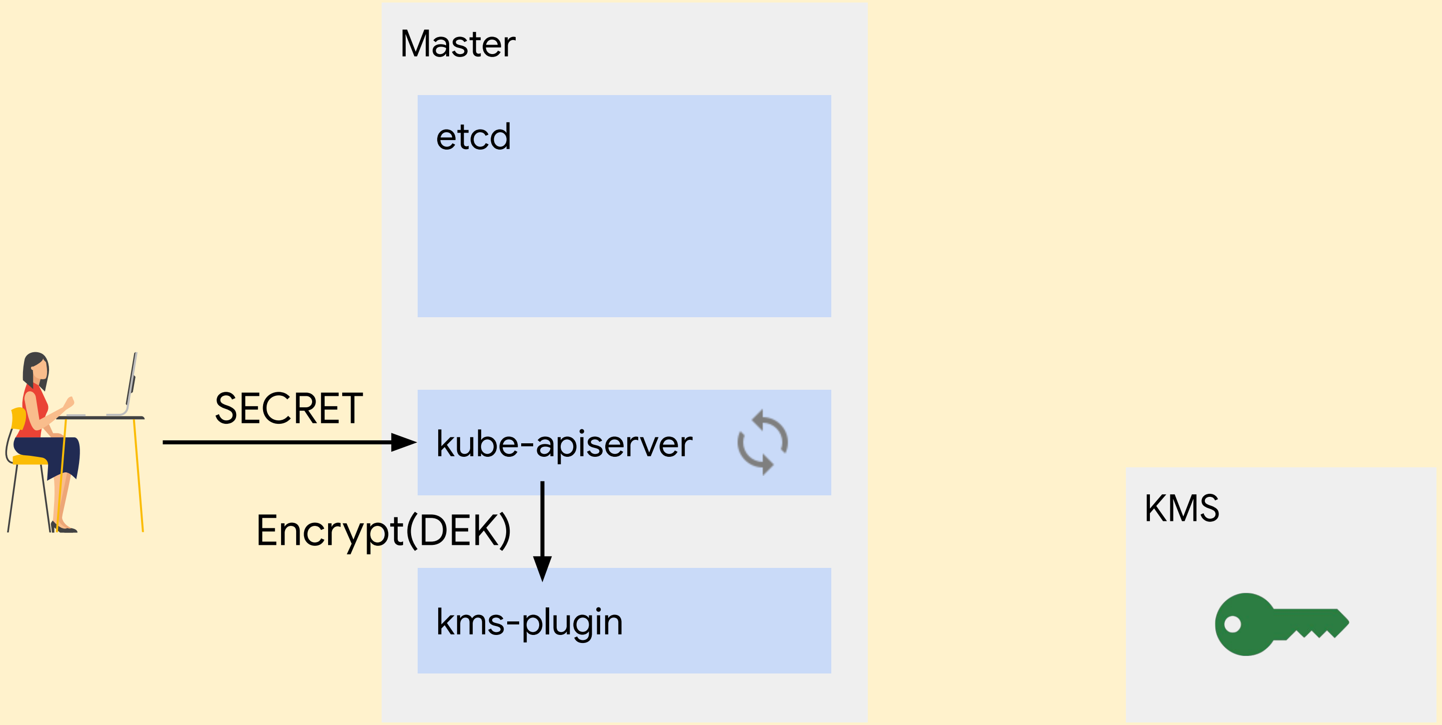
Envelope encryption sequence



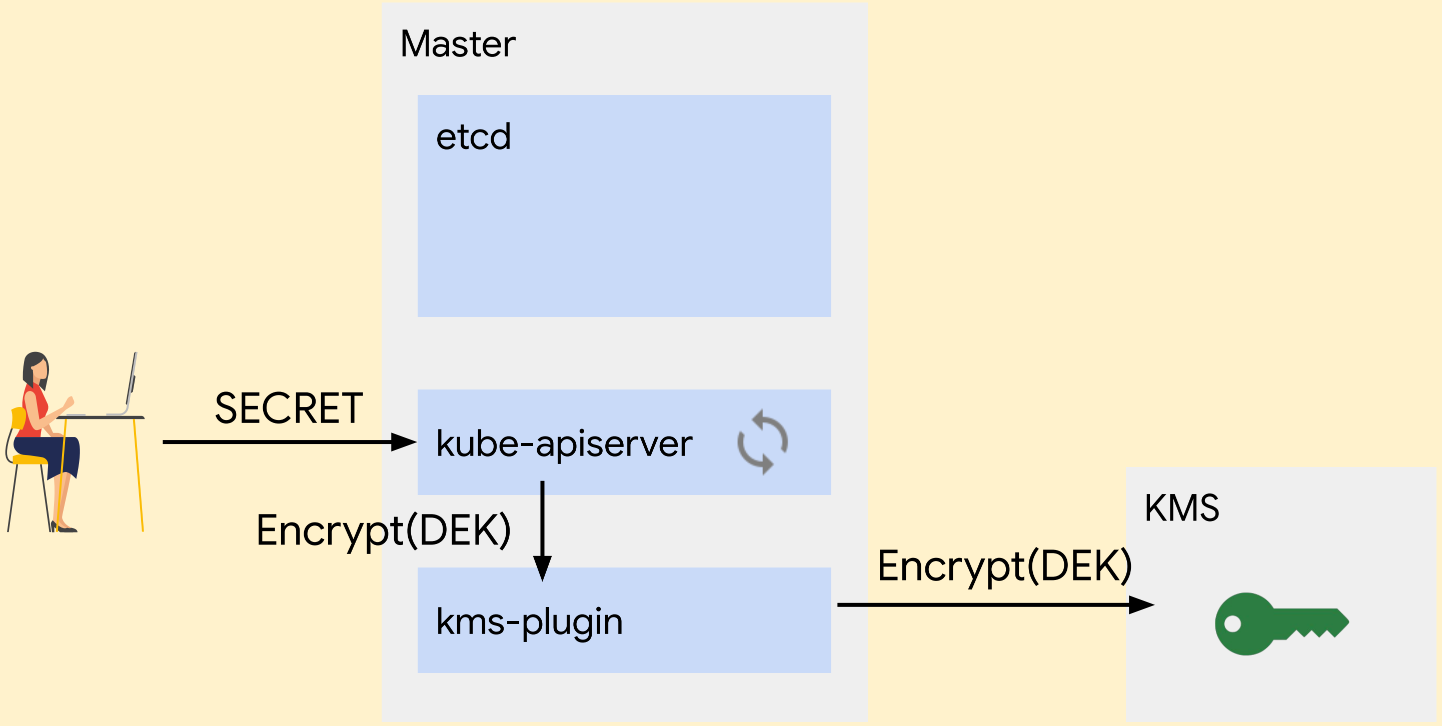
kube-apiserver generates a DEK



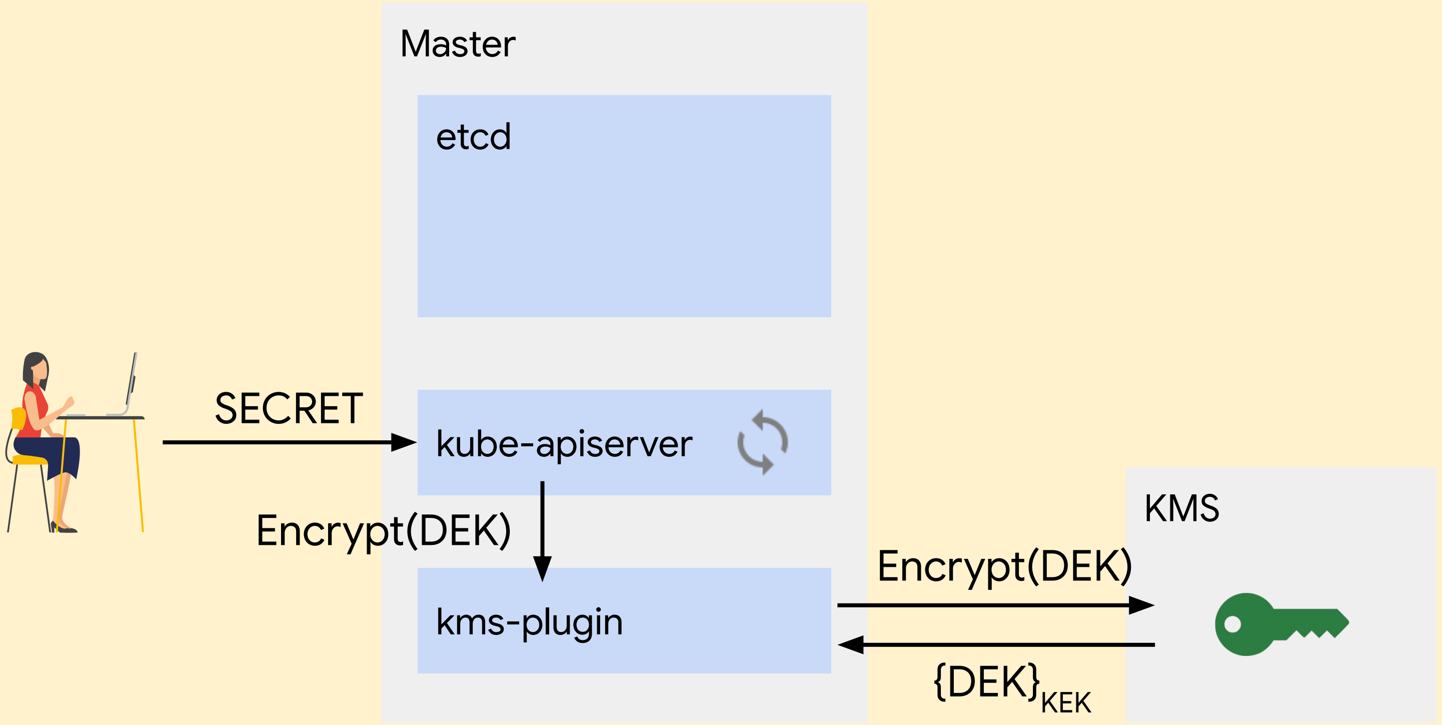
kube-apiserver sends DEK to kms-plugin



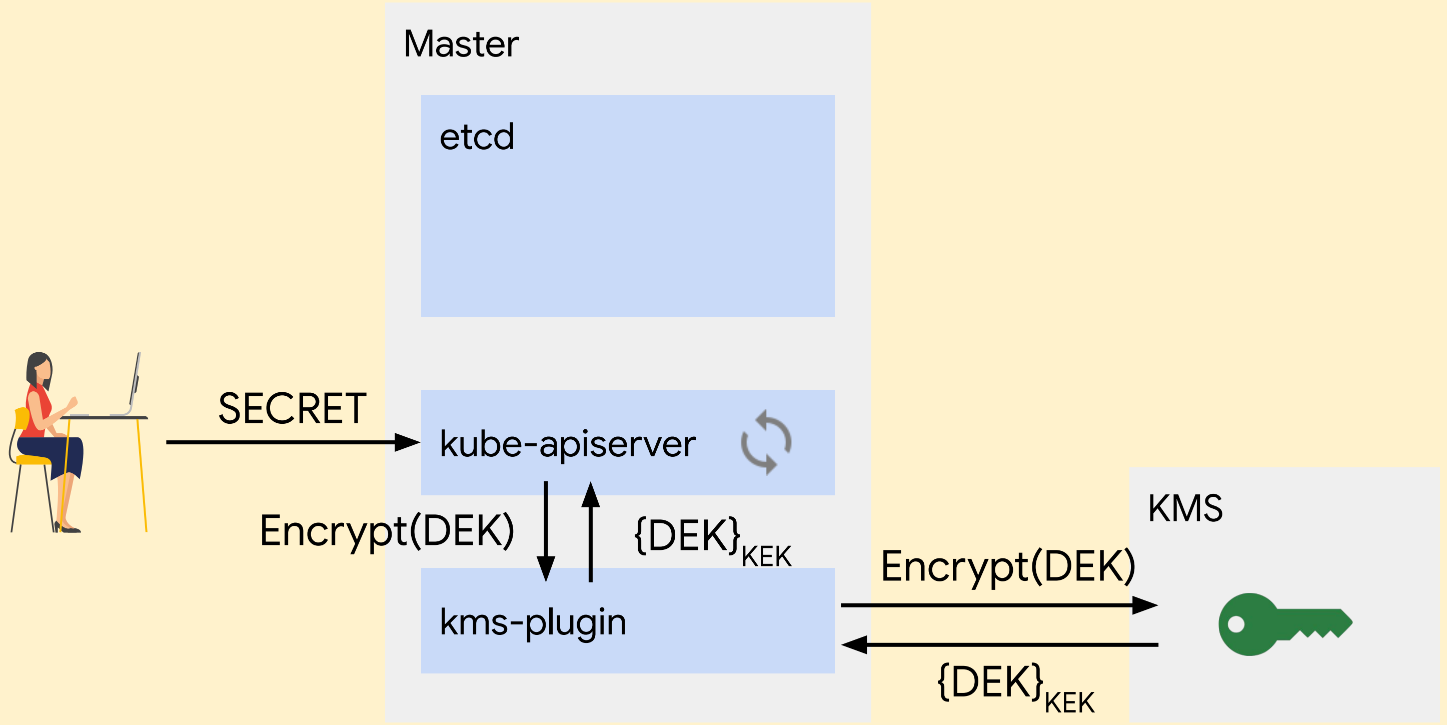
kms-plugin forwards to KMS



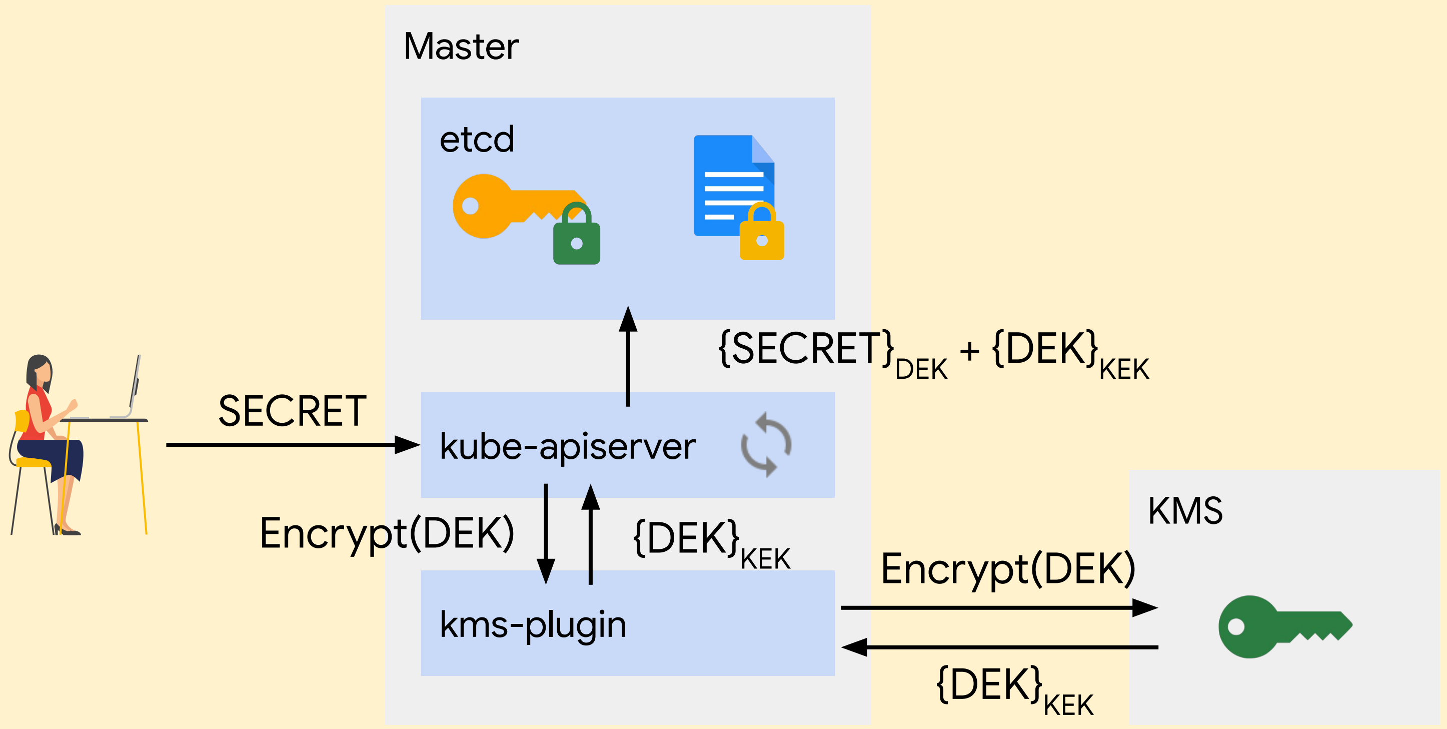
KMS encrypts a DEK



kube-apiserver constructs an envelope



enveloped Secret is saved to etcd



Step #6: add configurable DEK type to KMS plugin

k8s.io/apiserver/pkg/apis/config/types.go
k8s.io/apiserver/pkg/apis/config/v1/types.go

```
type KMSConfiguration struct {
        Name string
        CacheSize int32
        Endpoint string
        Timeout *metav1.Duration
        DEKType string
}
```

https://github.com/awly/kubernetes/commit/1b7fb37bad4ed4e717b14f16e0e26e52d3267662

Step #7: teach KMS plugin about your new DEK type

k8s.io/apiserver/pkg/server/options/encryptionconfig/config.go

```
func GetPrefixTransformers(config *apiserverconfig.ResourceConfiguration)
([]value.PrefixTransformer, error) {
    …
    if provider.KMS != nil {
        switch provider.KMS.DEKType {
        case "myType":
            newDEKTransformer = mytransformer.New
            dekSize = myKeySize
        }
    }
}
```

https://github.com/awly/kubernetes/commit/93cdb73c430795de4886648a97bd34e26d0c677b

Step #8: choose your KMS provider and plugin

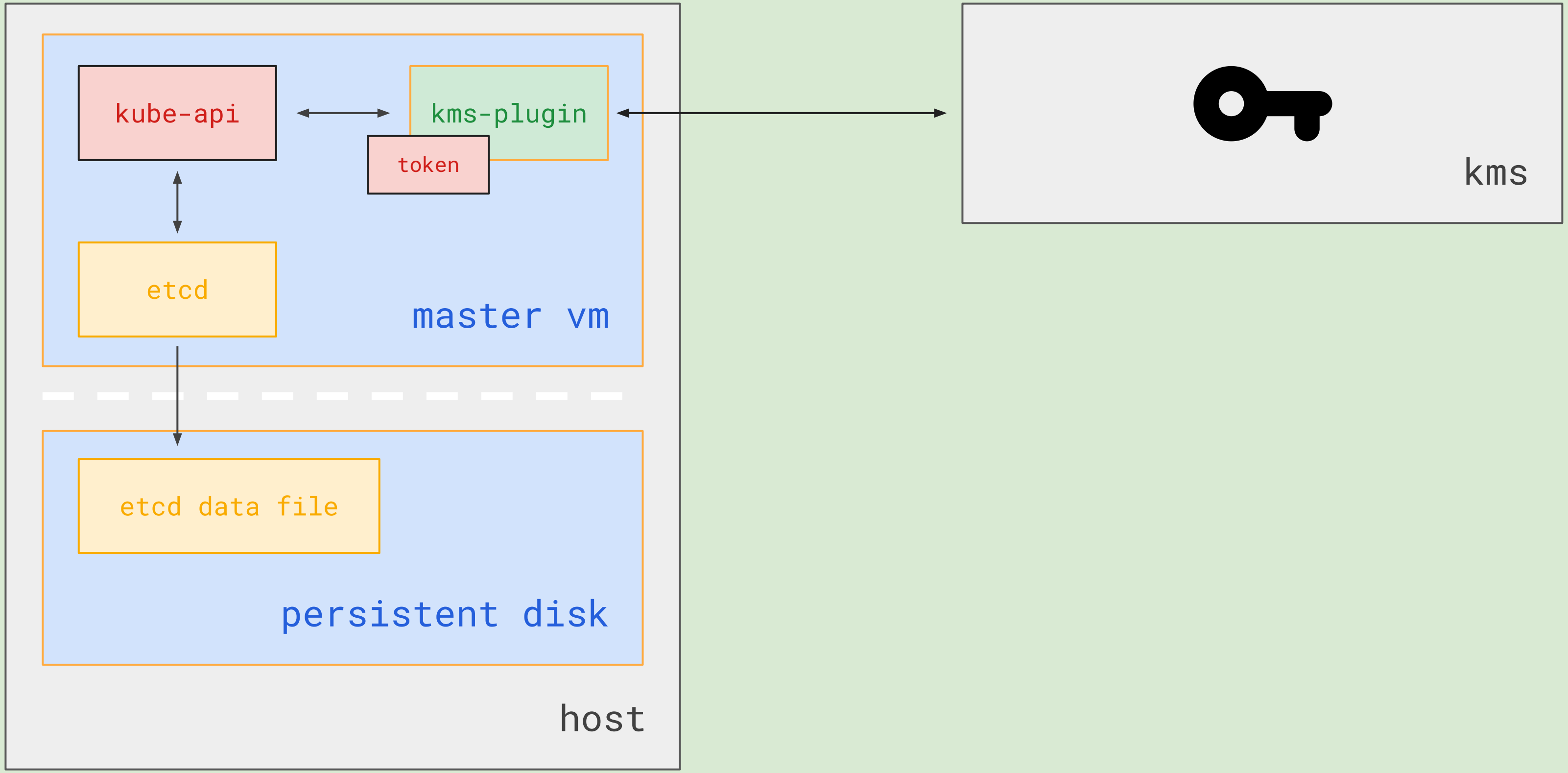
Google Cloud KMS: https://github.com/GoogleCloudPlatform/k8s-cloudkms-plugin/

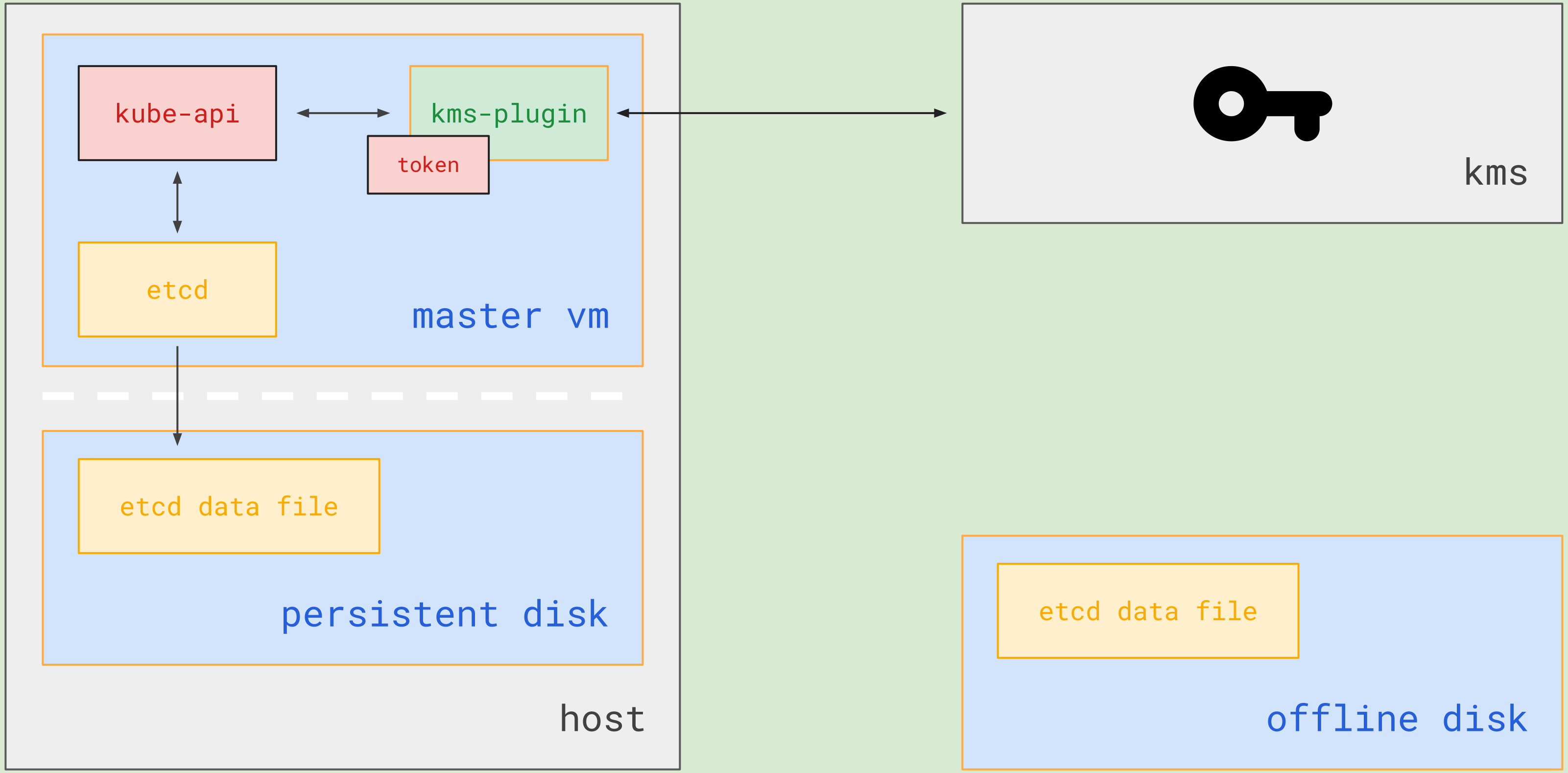
Microsoft Azure Key Vault: https://github.com/Azure/kubernetes-kms

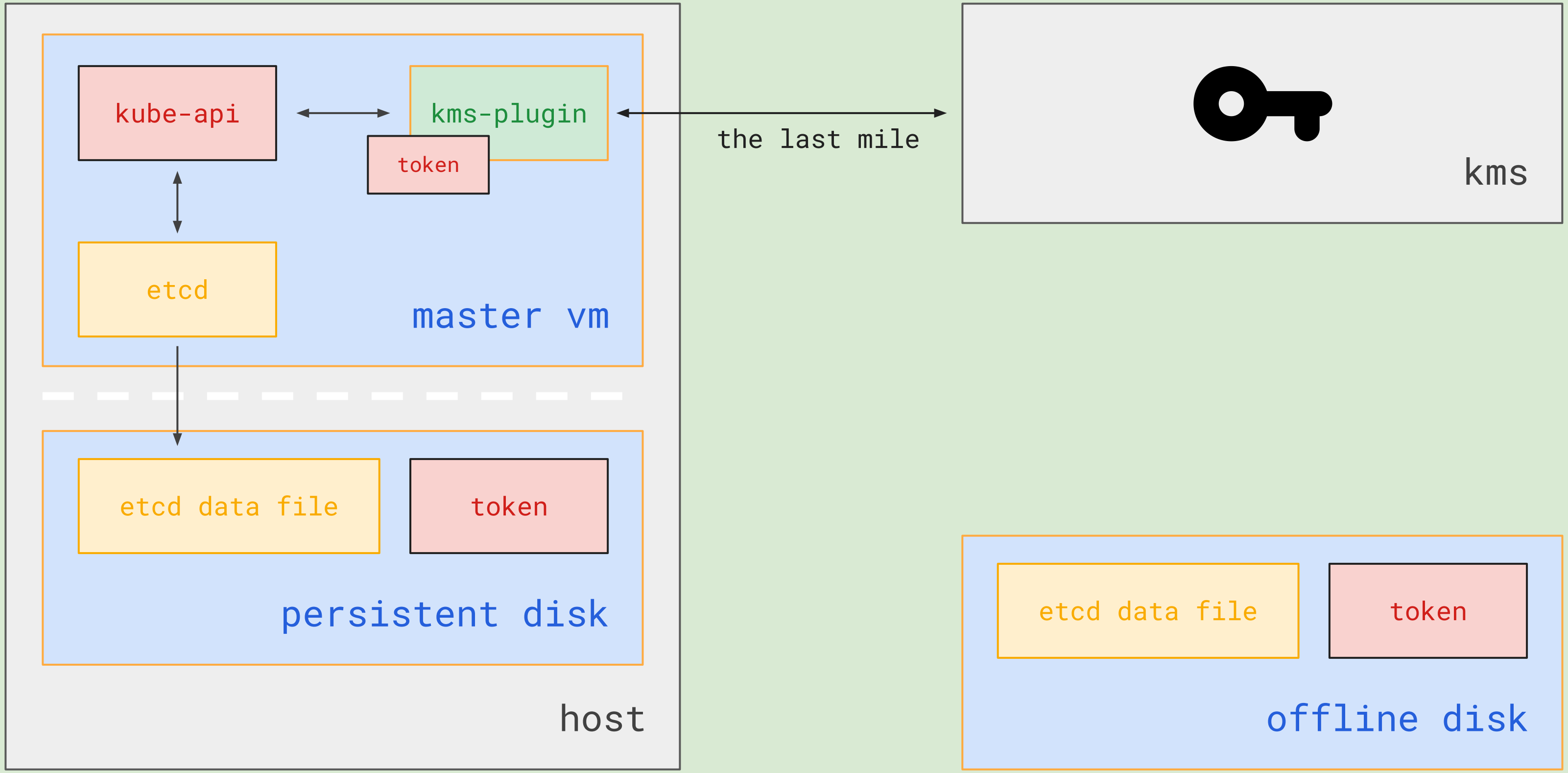
AWS KMS: https://github.com/kubernetes-sigs/aws-encryption-provider

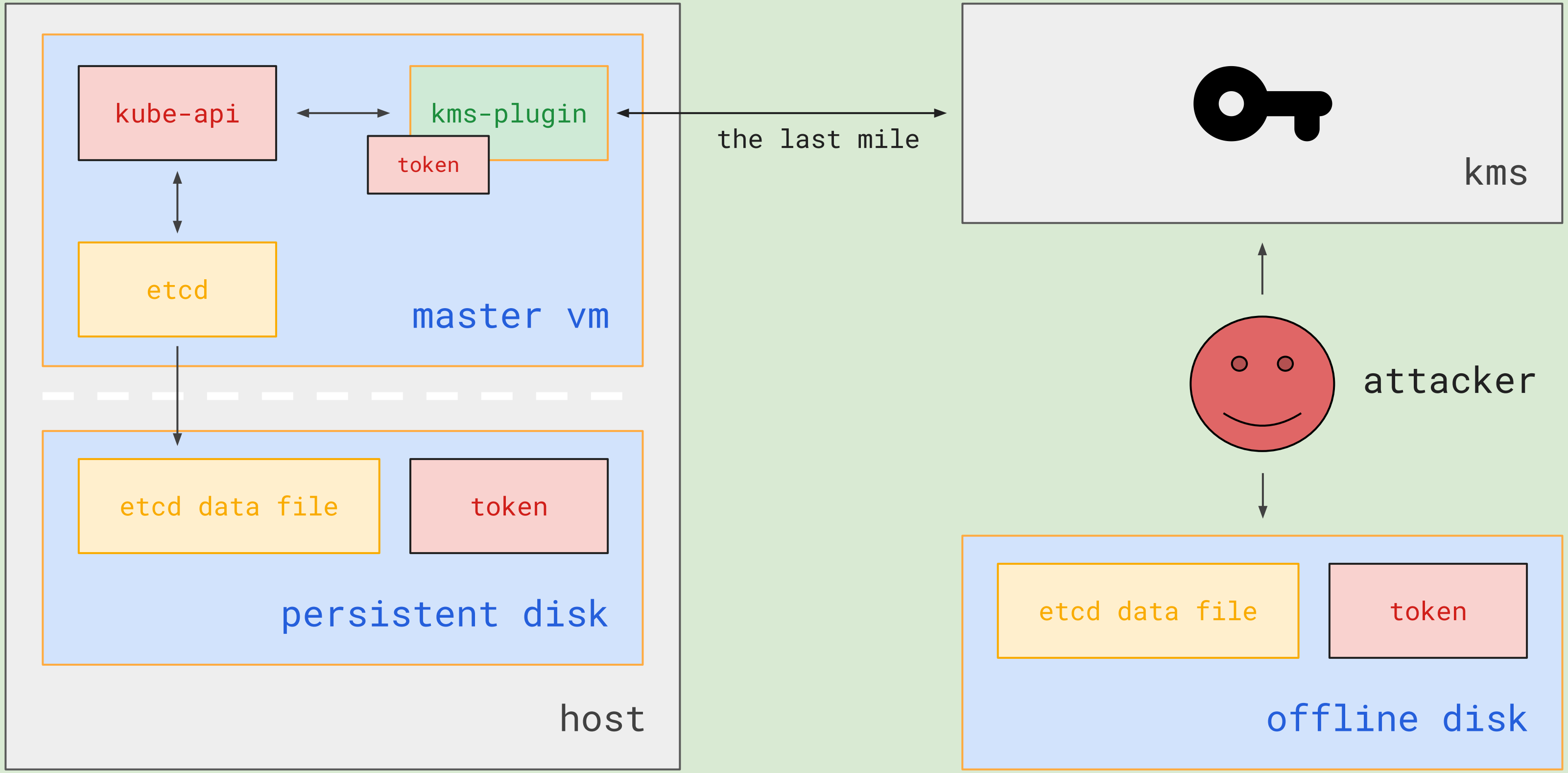
HashiCorp Vault: https://github.com/oracle/kubernetes-vault-kms-plugin

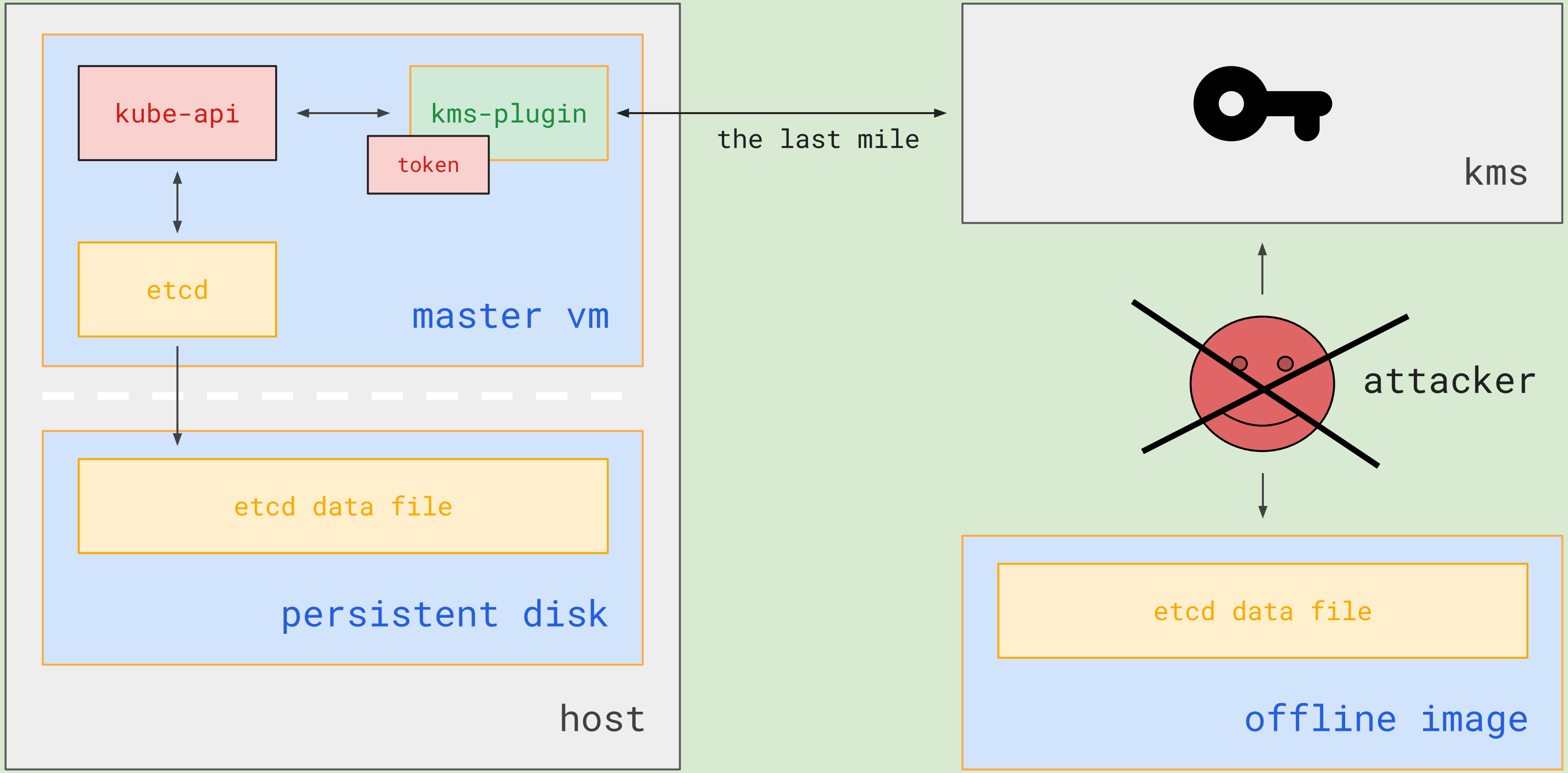
Threat Model of KMS Plugin


kube-api
kms-plugin
token
kms
etcd
master vm
etcd data file
persistent disk
host


kube-api
kms-plugin
token
kms
etcd
master vm
etcd data file
persistent disk
host
etcd data file
offline disk


kube-api
kms-plugin
token
the last mile
kms
etcd
master vm
etcd data file
token
persistent disk
host
etcd data file
token
offline disk


kube-api
kms-plugin
token
the last mile
kms
etcd
master vm
attacker
etcd data file
token
persistent disk
host
etcd data file
token
offline disk


kube-api
kms-plugin
token
the last mile
kms
etcd
master vm
attacker
etcd data file
persistent disk
etcd data file
host
offline image

Protecting KMS plugin with Trusted Platform Modules

What’s a Trusted Platform Module (TPM)?

Crypto coprocessor

Protected memory boundary, outside of kernel reach

Bound to the host machine

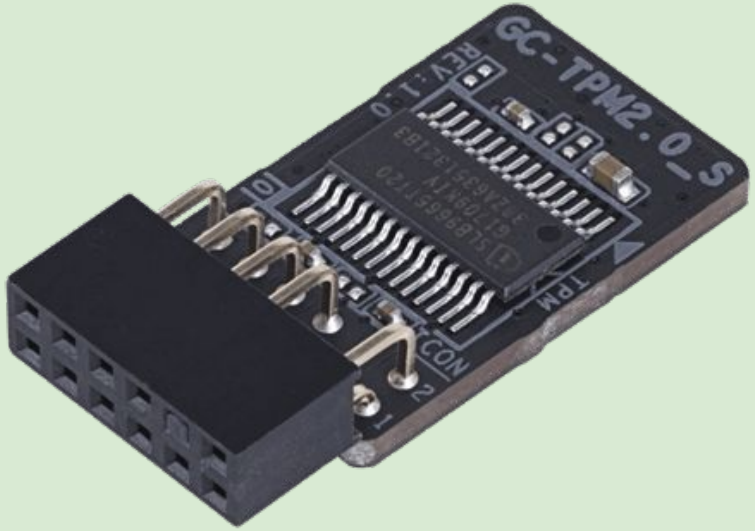
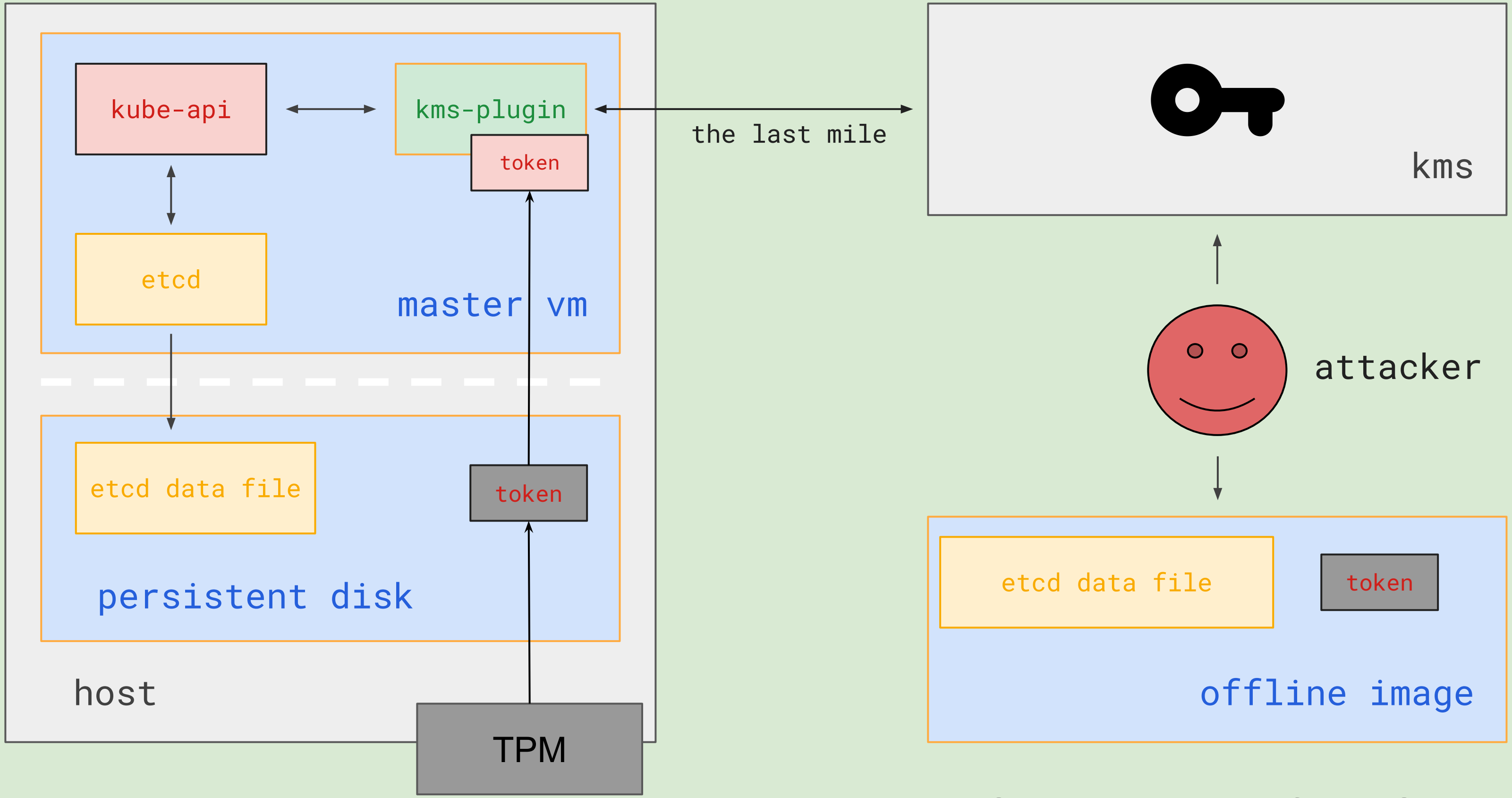
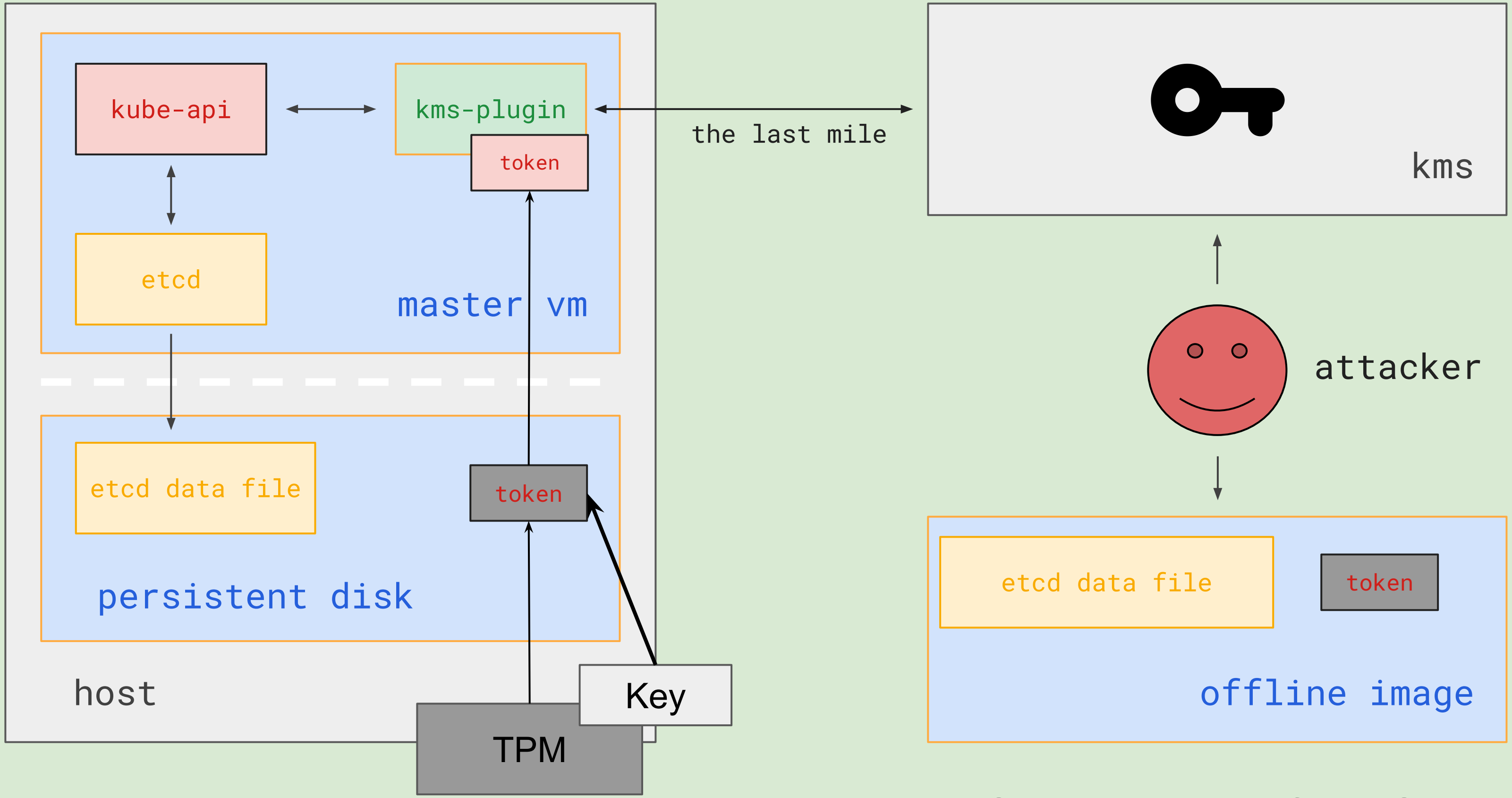


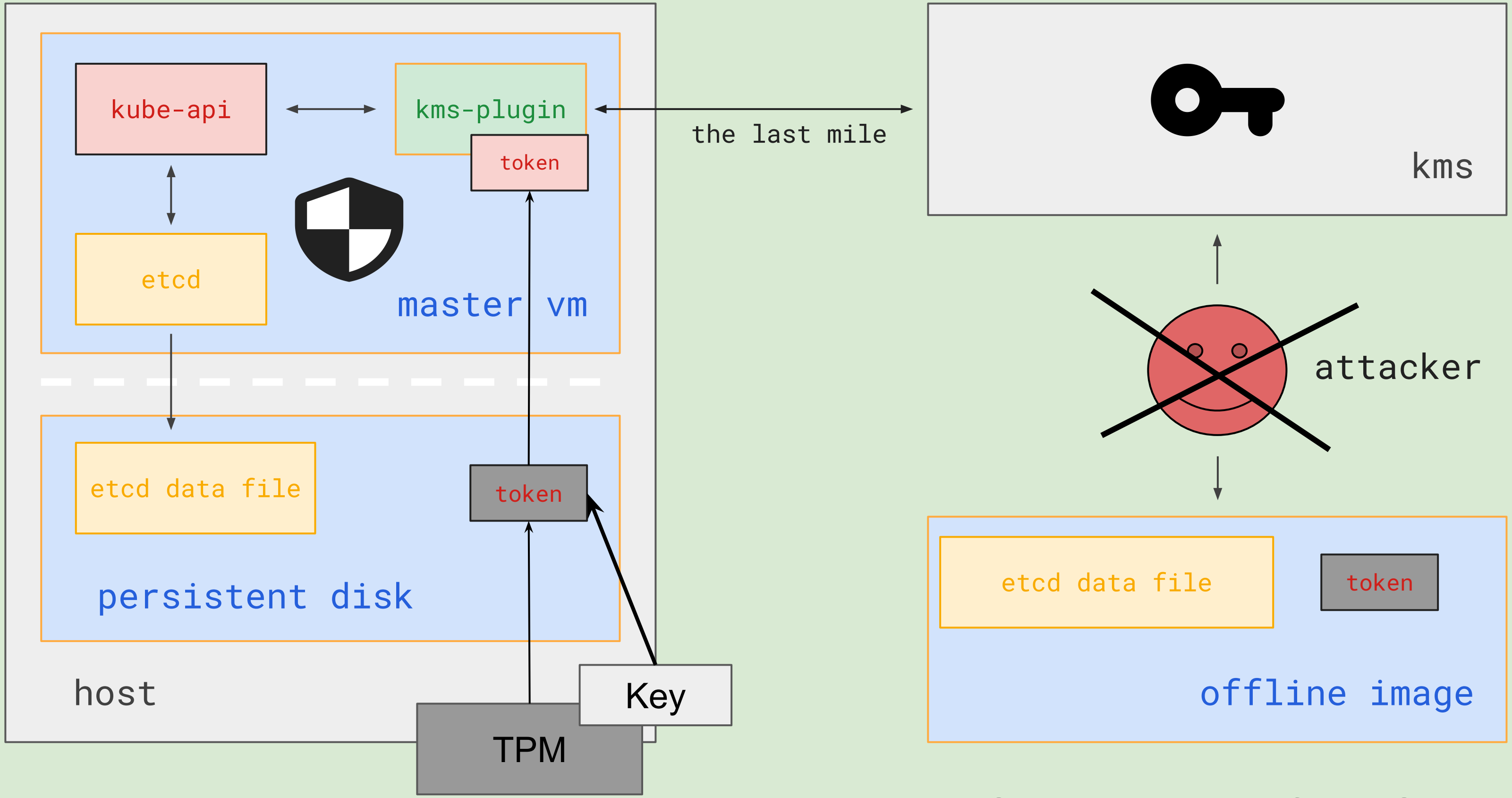
Image source: https://www.gigaparts.com/gigabyte-trusted-platform-module-2-0-gc-tpm2-0-s.html



Seal the credential for KMS in TPM



Seal the credential for KMS in TPM



Seal the credential for KMS in TPM

Summary

1. Transformers mutate data at etcd boudry
2. Layers of Transformers
   a. built-in storage transformer
   b. envelope transformer
   c. KMS plugin
3. TPMs for last-mile credential protection

Call to action

1. **Encrypt your Secrets at rest!**
2. Contribute to OSS!

References

1. Turtles All the Way Down, KubeCon China 2019
2. Securing Kubernetes with TPMs, KubeCon EU 2019
3. KMS plugin talk from Next 2019
4. Best practices for writing gRPC services

Backup slides