


# RFC: Merge istio-ca-secrets and cacerts

Shared with Istio Community

	
<b>Owner:</b> Jackie Elliott ( <a href="#">jaellio</a> ) <b>Working Group:</b> Security	<b>Status:</b> WIP   <b>In Review</b>   Approved   Obsolete <b>Created:</b> 2023-03-07 <b>Approvers:</b>

## TL;DR

The *istio-ca-secret* and *cacerts* Kubernetes Secrets offer alternative options to provide the root or intermediate certificate to istiod in different scenarios, but contain similar information. This doc proposes merging the *istio-ca-secret* and *cacerts* Kubernetes secrets into a single secret (surfaced via [Propose combine "istio-ca-secret" and "cacerts" · Issue #41507 · istio/istio \(github.com\)](#)).

## Background

Currently, Istio supports two mechanisms for configuring the CA certificate when Istiod is functioning as the Certificate Authority.

1. [Plugin CA certificate](#)
2. Istiod generated self-signed certificate

There are several functionality and configuration differences between the plugin *cacerts* and the *istio-ca-secret* self-signed certificate. The *cacerts* Secret is optionally volume mounted to istiod. Since Istio v1.12, Istiod has supported picking up on changes to the plugin CA certificate in the *cacerts* Secret without having to restart istiod: [Add support to istiod to notice cacerts changes by rveerama1 · Pull Request #31522 · istio/istio \(github.com\)](#). The generation of a self-signed CA certificate stored in the *istio-ca-secret* k8s Secrets is not recommended for production and istiod does not pick up on a change to *istio-ca-secret* without a restart. Istiod expects a *cacerts* Secret to be generic or tls certificate type. The *istio-ca-secret* k8s Secret is of type *istio.io/ca-root*.

## Goals

- Reduce complexity of providing a root certificate for the Istio CA and centralizing the location of a root certificate

- Merge the *cacerts* and *istio-ca-secret* into single k8s Secret for plug in and istio generated CAs
- Maintain backwards compatibility with *cacerts* and *istio-ca-secret* functionality

## Non-Goals

- Create a new root certificate workflow or means to provide an additional root certificate

## Requirements

To preserve backwards compatibility, the solution must preserve the existing functionality of the plugin CA cert as well as the self-signed certificate. The solution aims to preserve the functionality introduced in [Add support to istiod to notice cacerts changes by rveerama1 · Pull Request #31522 · istio/istio \(github.com\)](#). This change implemented a file watch on the volume mounted *cacerts* Secret. Currently, no watch is initialized on the *istio-ca-secret*.

The use cases of a merged CA root Secret are comprised of the combined use cases of the plugin CA and a self-signed, istiod generated CA certificate.

## Proposal/Design Ideas

Utilize the existing *cacert* for the plugin and istiod generated root certificate. The *cacerts* Secret is either created and populated by the user with a plugin certificate, or it is created by istiod and populated using a self-signed certificate. The supported types for *cacert* are generic, *istio.io/ca-root*, and *tls* certificate.

The *cacerts* Secret will remain as an optional Secret for user provided certificates.

## Startup Workflow

On start-up, istiod will follow the below process to load or create the CA certificate:

1. First, istiod checks for the existence of a file mounted *cacerts* Secret by checking for the populated well known file paths.
  - a. If the files at the well known path are not empty, the existing workflow will continue as is.
  - b. *NOTE: This workflow is already implemented.*
  - c. If the Secret does not exist, istiod will check for the existence of the *istio-ca-secret*.

2. Second, istiod checks for the existence of the *istio-ca-secret*.
  - a. If the secret exists, istiod creates a *cacerts* Secret of the same Secret type and populates the Secret with the contents of *istio-ca-secret*. If *cacerts* already exists, ignore the error. Then a watch is established on the *cacerts* [optional Secret volume mount](#).
    - i. Note: This enables supports for updating the istio generated self-signed without having to restart the control plane.
  - b. If the secret does not exist, istiod creates a *cacerts* Secret.
3. Lastly, istiod generates a self-signed cert as its CA.
  - a. Istiod attempts to create the *cacerts* Secert as type istio.io/ca-root and populate with the new self-signed cert. A file watch is initialized on the optional Secret volume mount.
  - b. If the Secret creation fails because the secret already exists then a watch is established on the existing secret.

## Alternative Workflow (Draft)

1. Create new *cacerts* file watch process only for the initial load
  - a. Sends a cancel context if the following operation is no longer needed
2. Check for *cacerts* files and load them if present. Start existing *cacerts* file-watcher and continue with existing workflow. Cancels the above file watch
3. Attempt to get istio-ca-secret in a retry loop - breaks after 20 minutes? idk? similar to existing code
  - a. If operation fails does to a not found error, create a self signed cert and populate a new *cacerts* Secert - if that fails error, unless the error is that is already exists then break
  - b. If operation fails with another error, retry *istio-ca-secret* get operation
  - c. If found, create a *cacerts* from the contents of *istio-ca-secret* - if that fails error, unless the error is that is already exists then break
4. Either the *cacerts* file watch finds the files or the watch is cancelled after 20 minutes and errors
5. Load the *cacerts* file and start the existing watch functionality

## Challenges Encountered

When a istiod generates a self-signed root certificate for use by the internal CA, a self-signed ca root certificate rotator is initialize. The *SelfSignedCARootCertRotator* automatically checks self-signed signed root certificates and rotates the root certificate if it is going to expire. The rotator watches the istio-ca-secret k8s secret via the *caSecretController*.

*checkAndRotateRootCert()* loads the istio-ca-secret calls

*checkAndRotateRootCertForSigningCertCitadel()*. The method gets the wait time until the

certificate needs to be rotated. Even if the certificate doesn't need to be rotated, the method checks if the certificate bytes in the secret are different from the ca cert bytes in the local CAKeyCertBundle (indicating another instance rotated the cace. If it is different, the CAKeyCertBundle is updated - the ca cert and ca private key are overwritten and the previous rootCertFile is appended to the new ca cert and set to the root cert of the key bundle.

If the secret did need to be rotated, then the method gets the existing options from the istio-ca-secret ca-cert.pem. The options are merged with the new options which include the SignerPrivPem (this is the same as the private key of the old cert. The new root is generated from the existing private key.

*UpdateRootcertificate()* updates the root cert in istio-ca-secret, the keyCertBundle, and the configMap. The method accepts an argument to either roll forward or backwards a rotation.

When a Istiod starts up, identifies a volume mounted secret cacerts, and loads the secret data, the file watch on the cert data is initialized. When the files are updated the CAKeyCertBundle is updated. *updatePluggedinRootCertAndGenKeyCert* is called on an update and istiodCertBundleWatcher is notified.

The *istiodCertBundleWatcher* periodically compares it's own version of the CAKeyCertBundle to the existing CAKeyCertBundle. If they are not equal the server's *bundle* is set to the new value and the *istiodCertBundleWatcher* is notified.

## **Problem**

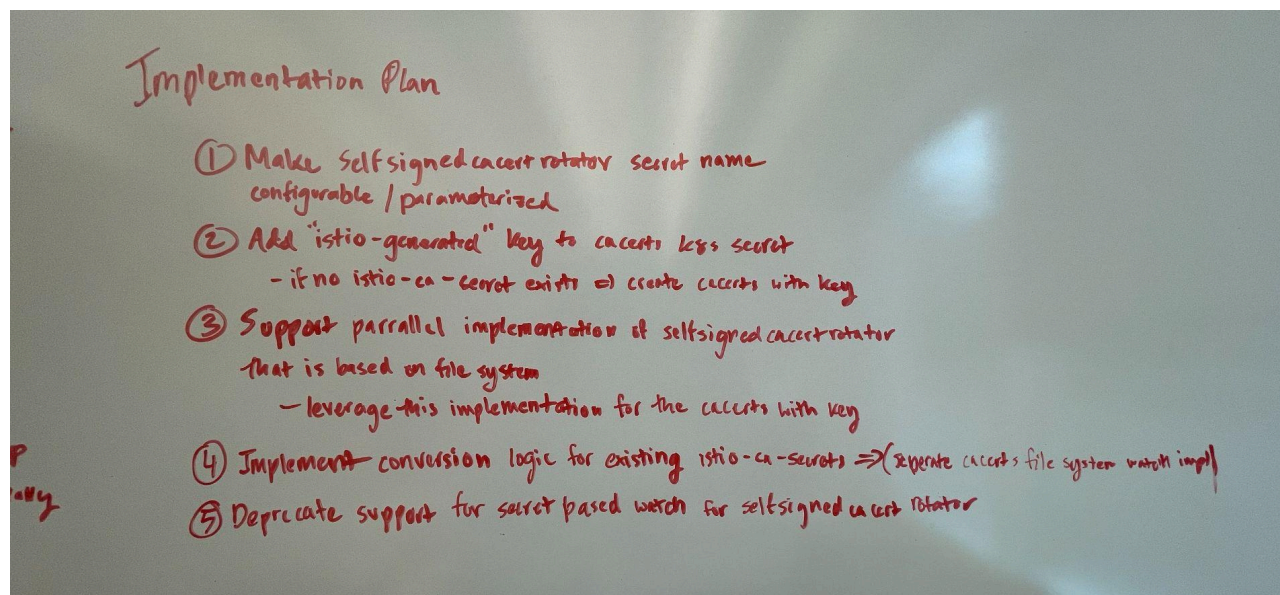
Depending on whether or not a plugged in certificate or an istiod generated self-signed certificate is used for the CA, different watch and update mechanisms are initiated. With the proposed secret merging, there is currently no way for all istiod instances to determine if the *cacerts* secret was generated by Istiod or not. Therefore, the existing self-signed ca root cert rotator mechanism might not be initialized correctly or the user might not be aware that the secret will be rotated/updated by istiod. Additionally, the self-signed ca root cert rotator depends on a secret watch rather than a file watch. The goal of the proposal is to simplify the CA configuration process by merging the istio-ca-secret and cacerts k8s secrets, but with the above mentioned problems the backend implementations would be vastly different.

One possible solution is to update the *SelfSignedCARootCertRotator* to rely on a file watch (similar to the existing watch for plugged in ca certs) rather than a secret watch on istio-ca-secrets. The rotator will periodically check the file contents for expiration rather than the Secret itself. If the cert needs to be rotated, the secret itself will be updated. The change will eventually be picked up via the file system watch and the current plugged in cert mechanisms will continue. The remaining challenge is how to distinguish between istiod generated self-signed and plugged in certificates when we are only examining the file system.

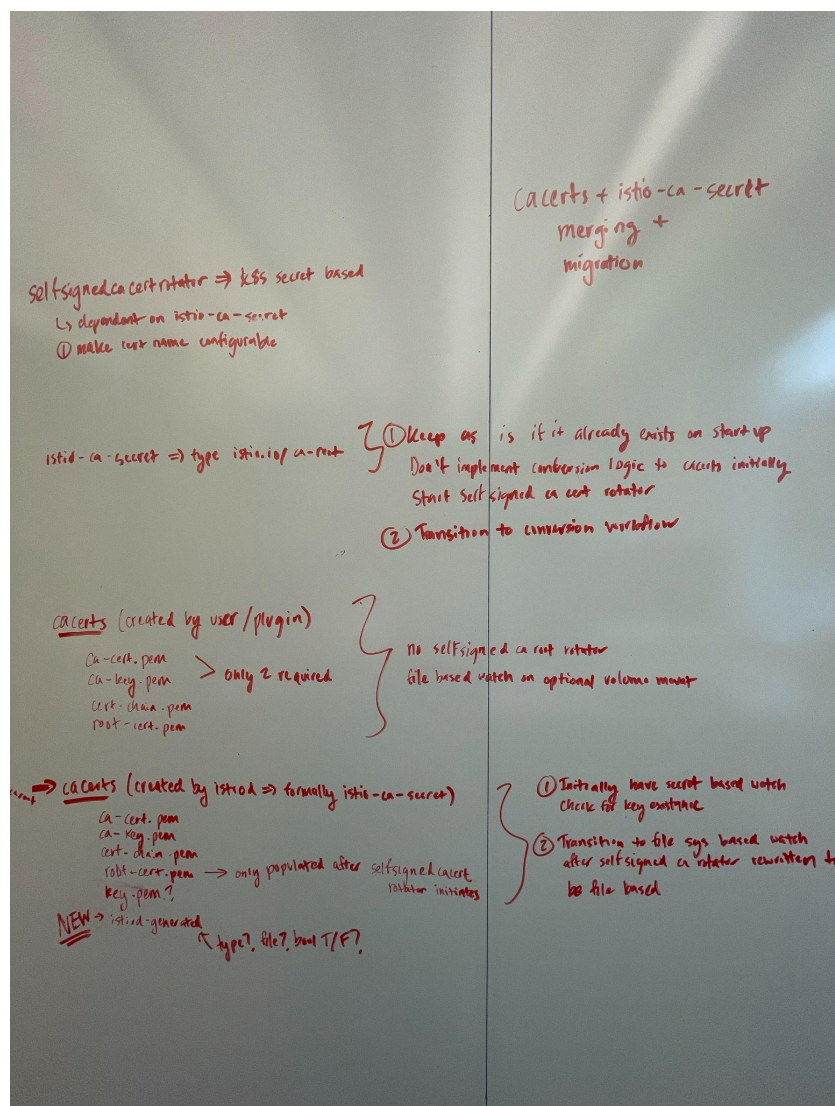
## Alternative Options

- Enable the user to specify the plugin CA secret name on install. Currently, the plugin CA Secret must be named *cacerts*. This configurability would give user more freedom over what secrets they can plugin to istiod and would require less reconfiguration.
  - Downsides: Additional complexity, challenging debugging scenarios, inconsistent environments, doesn't simplify CA secret options
  - Benefits: User configurability
- [ACCEPTED APPROACH] Use the existing *cacerts* Secret as the self-signed and plugin CA. This option proposes combining the *cacerts* and *istio-ca-secret* into the existing *cacerts*. On start-up, if a plugin *cacerts* did not exist istiod could create one with a self-signed CA. This change would require changes to existing logic rather than adding additional logic to handle a Secret of a new name.
  - Downsides: Expands the existing meaning of the *cacerts* Secret
  - Benefits: Doesn't require users to change any existing mechanism for updating *cacerts*, doesn't introduce a new secret, istio-generated self-signed *cacerts* and plugin *cacerts* changes will be picked up by the control plane without restart

## Implementation Plan







## Appendix

### Current cacerts Secret Configuration

Note: The cacerts secret can be of type generic or tls certificate.

kind: Secret

apiVersion: v1

metadata:

name: cacerts

namespace: istio-system

Type: Opaque

```
data:
  ca-cert.pem: <<ca-cert>>
  ca-key.pem: <<ca-cert-private-key>>
  cert-chain.pem: <<cert-chain>>
  root-cert.pem: <<root-cert>>
```

```
kind: Secret
apiVersion: v1
metadata:
  name: cacerts
  namespace: istio-system
type: kubernetes.io/tls
data:
  tls.crt: <<root-cert>>
  tls.key: <<root-cert-private-key>>
  ca.crt: ""
```

## ***Current istio-ca-secret Secret Configuration***

Note:

- The istio-ca-secret is of type istio.io/ca-root
- cert-chain.pem, key.pem, root-cert.pem are unset

```
kind: Secret
apiVersion: v1
metadata:
  name: istio-ca-secret
  namespace: istio-system
type: istio.io/ca-root
data:
  ca-cert.pem: <<root-cert>>
  ca-key.pem: <<root-cert-private-key>>
  cert-chain.pem: ""
  key.pem: ""
  root-cert.pem: ""
```