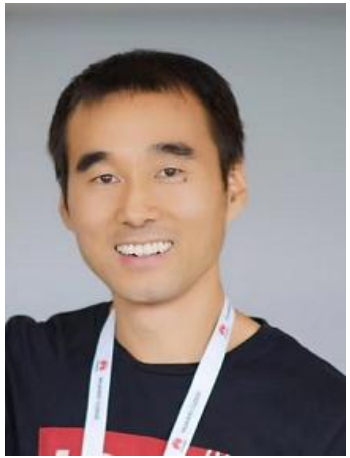
cert-manager Help Enhance Security and Flexibility of Istio Certificate Management

zhangchaomeng@huawei.com

About me



Chaomeng Zhang, Architect of Huawei Cloud Application Service Mesh, Architect of Distributed Cloud Native.

Chaomeng has cloud computing related design and developing work experience in Huawei Cloud, including service mesh, Kubernetes, micro service, cloud service catalog, big data, APM, cloud computing reliability and DevOps. He is Istio Steering Committee member, Istio community member, an experienced speaker of KubeCon, IstioCon, ServiceMeshCon, author of books "Cloud Native Service Mesh Istio"(《云原生服务网格 Istio：原理、实践、架构与源码解析》) and "Istio: the Definitive Guide"(《Istio权威指南》上册&下册)

Agenda

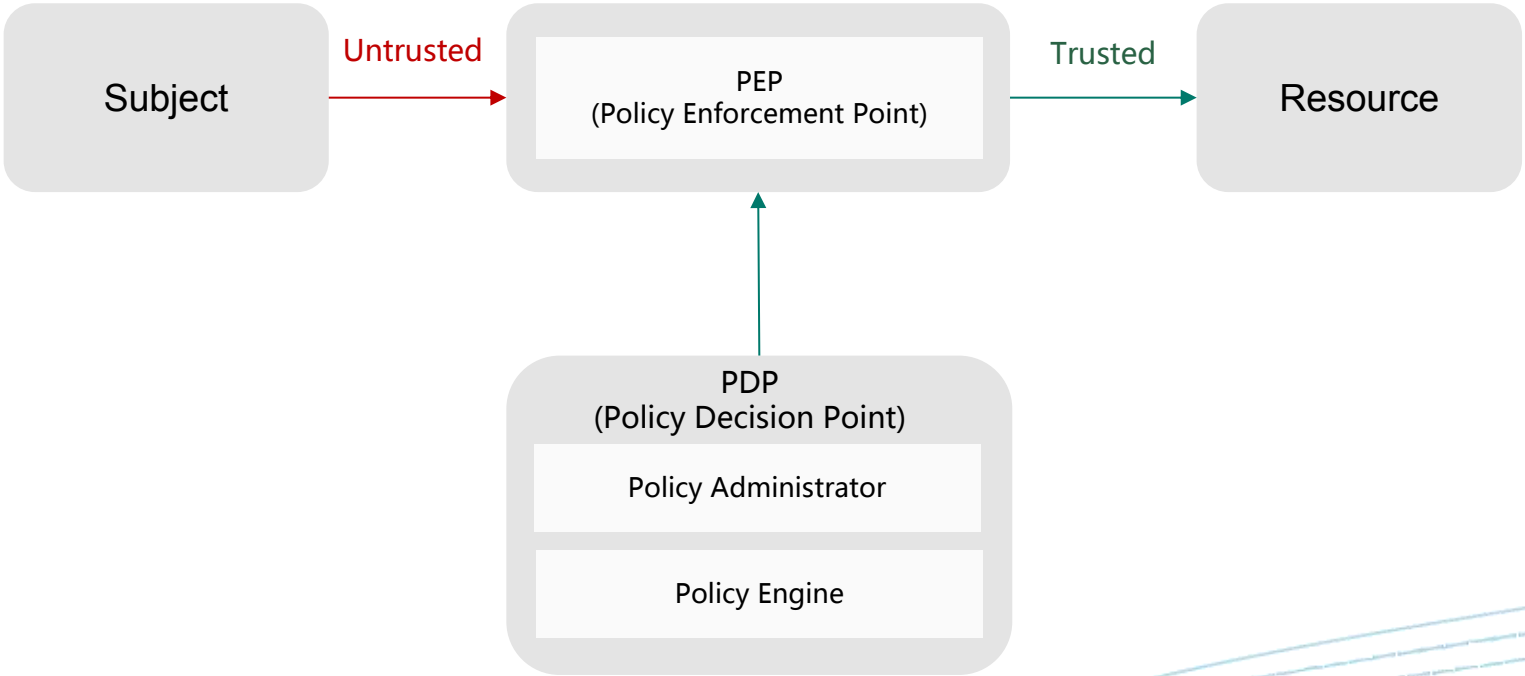
- Background: Istio Zero Trust Security
- Challenge: Certificate Management In Istio
- Solution: cert-manager
- Practice: cert-manager Issues Certificate For Mesh Root CA
- Practice: cert-manager Issues Certificate For Ingress Gateway

Zero Trust Security Model

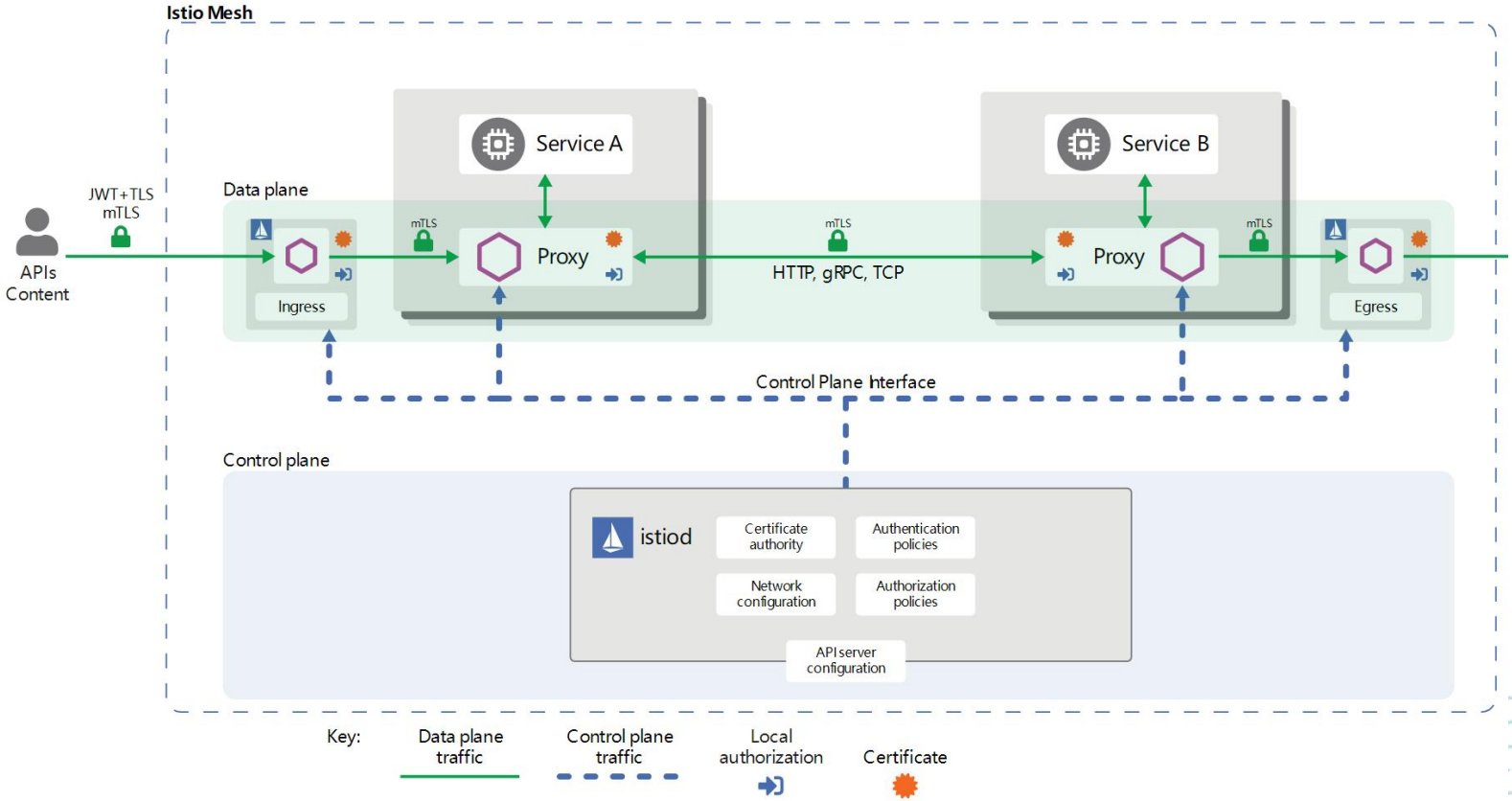
https://en.wikipedia.org/wiki/Zero_trust_security_model

- Describes an approach to **the strategy, design and implementation** of IT systems.
- The main concept is **"never trust, always verify"** , Users and devices **should not be trusted by default**, even if they are connected to a permissioned network such as a corporate LAN and even if they were previously verified.
- Implemented by establishing **strong identity verification**, **validating** device compliance prior to **granting access**, and ensuring **least privilege access** to only **explicitly authorized resources**.

Zero Trust Security Architecture Implementing



Istio Security Architecture

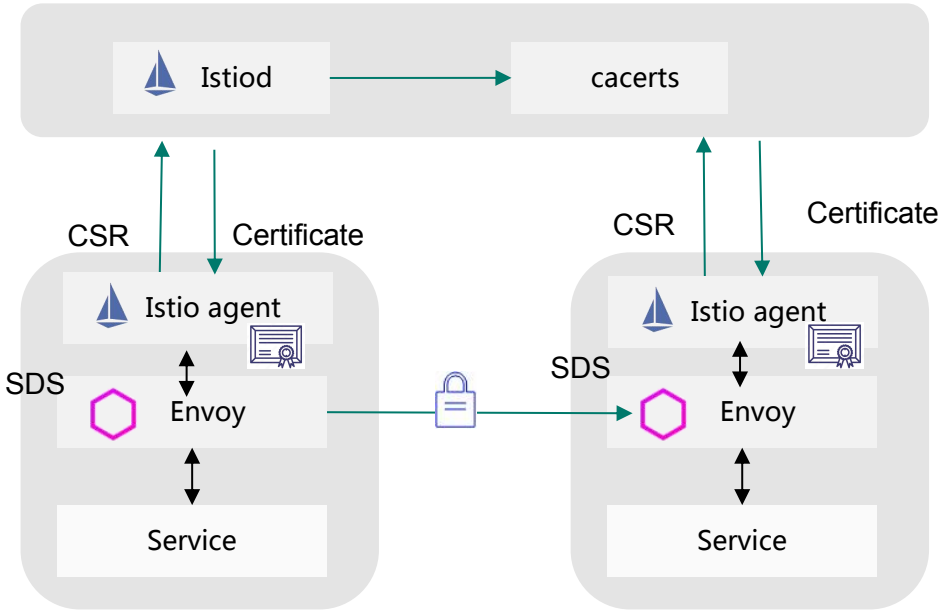


from Istio / Security

Agenda

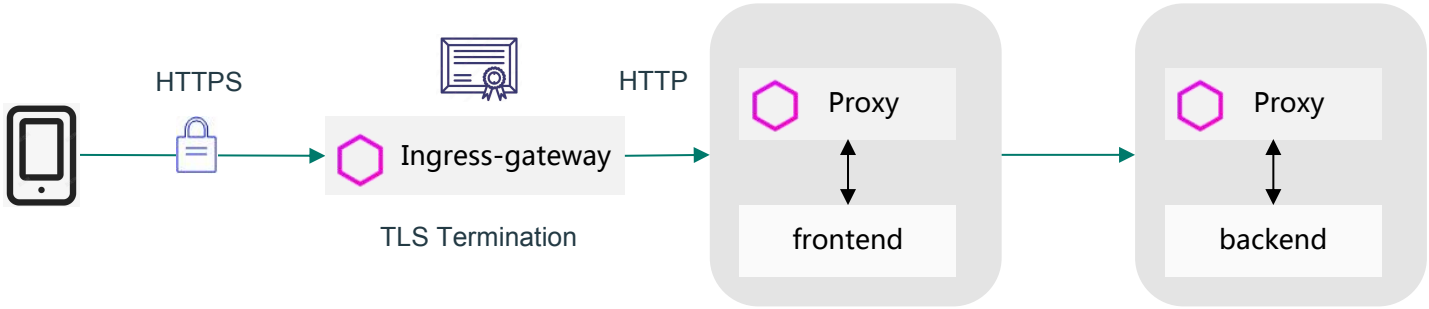
- Background: Istio Zero Trust Security
- Challenge: Certificate Management In Istio
- Solution: cert-manager
- Practice: cert-manager Issues Certificate For Mesh Root CA
- Practice: cert-manager Issues Certificate For Ingress Gateway

Certificate Inside: Transparent mTLS



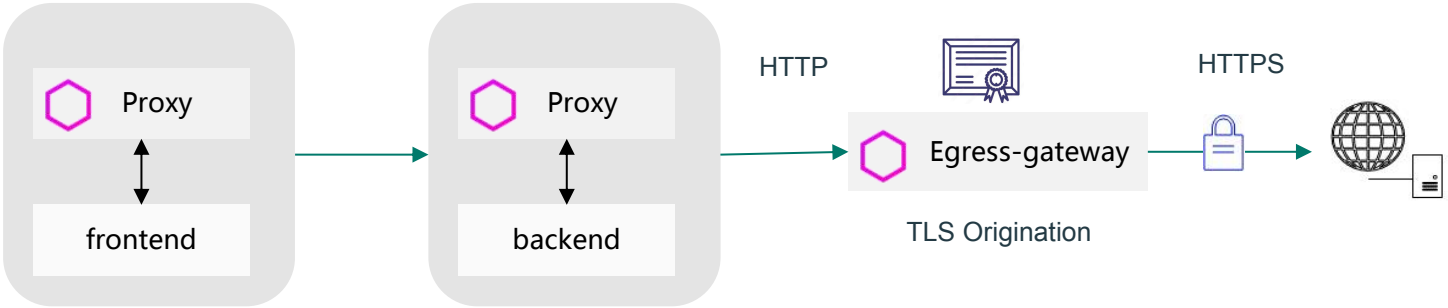
Istiod's certificate management helps singe and rotate workload certificate. But where does mesh root CA come from? And how to manage the root CA?

Certificate At Mesh Edge: Ingress TLS Termination



An ingress proxy accepts incoming TLS connections, decrypts the TLS, and passes unencrypted requests on to internal mesh services.

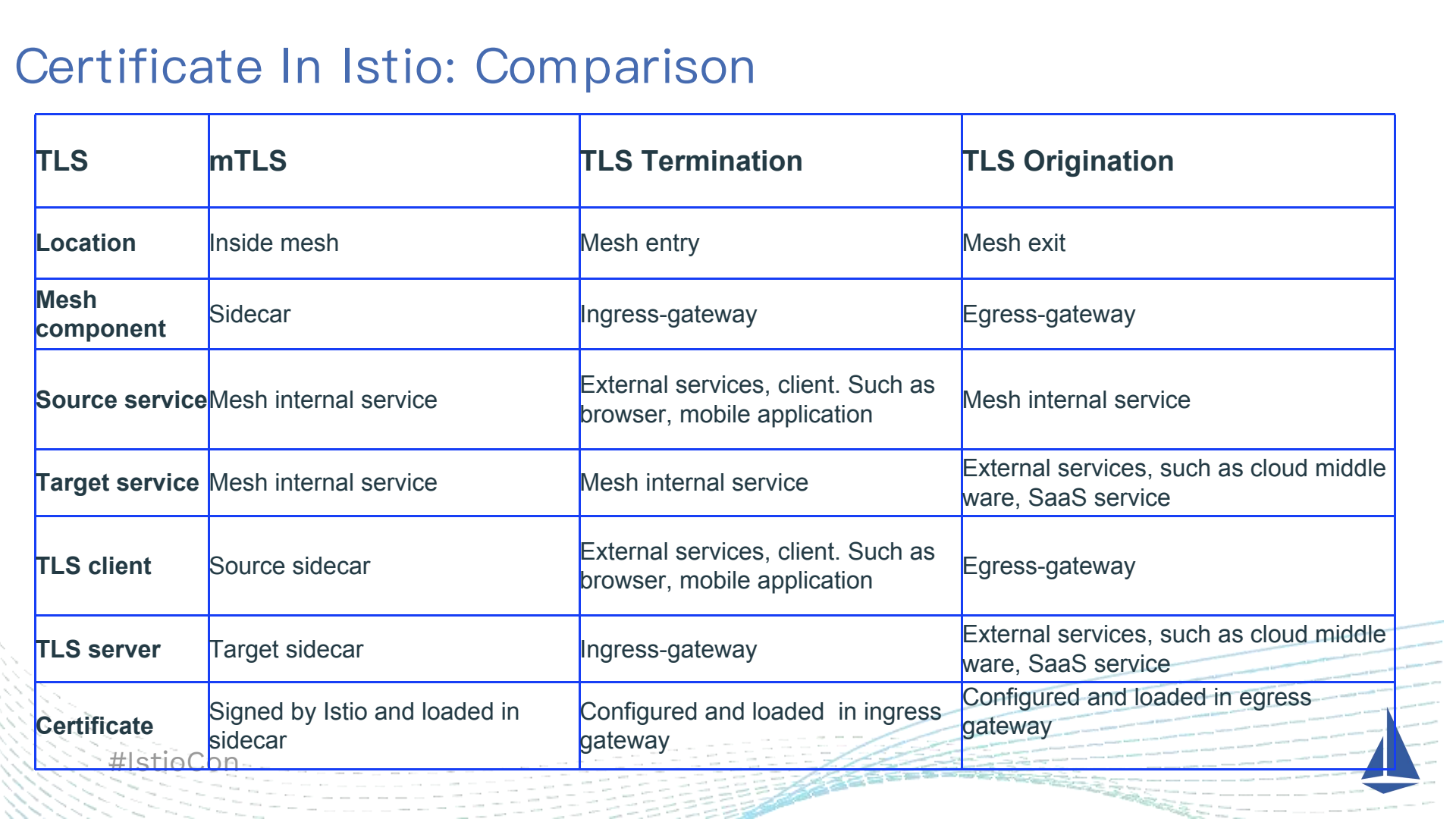
Certificate At Mesh Edge: Egress TLS Origination



TLS origination occurs when an Istio proxy (sidecar or egress gateway) is configured to accept unencrypted internal HTTP connections, encrypt the requests, and then forward them to HTTPS servers that are secured using simple or mutual TLS.

Certificate In Istio: Comparison

TLS	mTLS	TLS Termination	TLS Origination
Location	Inside mesh	Mesh entry	Mesh exit
Mesh component	Sidecar	Ingress-gateway	Egress-gateway
Source service	Mesh internal service	External services, client. Such as browser, mobile application	Mesh internal service
Target service	Mesh internal service	Mesh internal service	External services, such as cloud middle ware, SaaS service
TLS client	Source sidecar	External services, client. Such as browser, mobile application	Egress-gateway
TLS server	Target sidecar	Ingress-gateway	External services, such as cloud middle ware, SaaS service
Certificate	Signed by Istio and loaded in sidecar	Configured and loaded in ingress gateway	Configured and loaded in egress gateway



Challenges And Requirements:

- Avoid any downtime caused by certificates(lots of certificates problems in production environment)
- Automatically certificate renew before expiry(according to configured certificate duration and renew time)
- Flexibility of certificate configuration
- Rich issuer supported(public and private Certificate Authorities)
- Easy cloud native integrated

Agenda

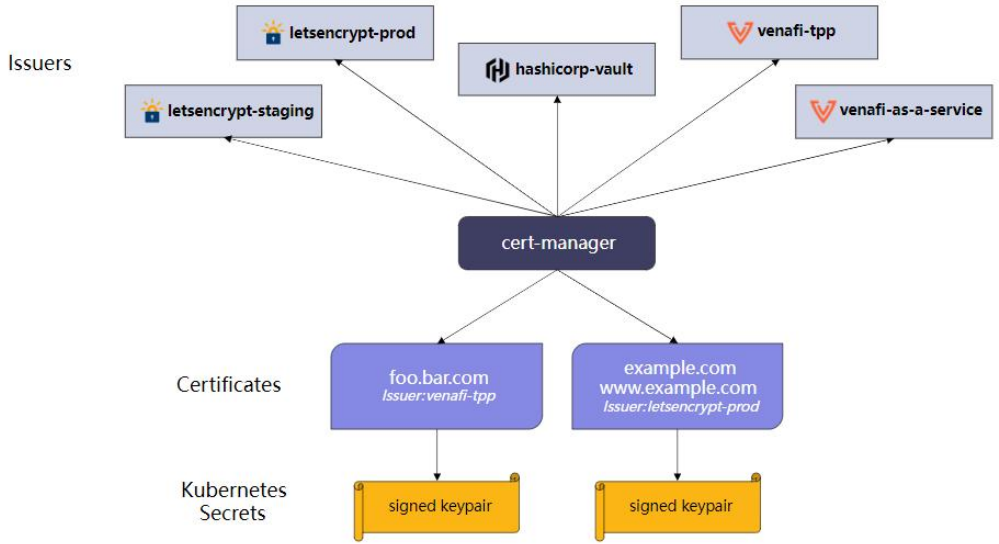
- Background: Istio Zero Trust Security
- Challenge: Certificate Management In Istio
- Solution: cert-manager
- Practice: cert-manager Issues Certificate For Mesh Root CA
- Practice: cert-manager Issues Certificate For Ingress Gateway

About cert–manager

cert-manager is a powerful and extensible X.509 certificate controller for Kubernetes and OpenShift workloads. It will obtain certificates from a variety of Issuers, both popular public Issuers as well as private Issuers, and ensure the certificates are valid **and up-to-date**, and will attempt **to renew certificates** at a configured time before expiry.

- Automated issuance and renewal of certificates to secure Ingress with TLS
- Fully integrated Issuers from recognised public and private Certificate Authorities
- Secure pod-to pod communication with mTLS using private PKI Issuers
- Supports certificate use cases for web facing and internal workloads
- Open source add-ons for enhanced cloud native service mesh security
- Backed by major cloud service providers and distributions

cert-manager Architecture



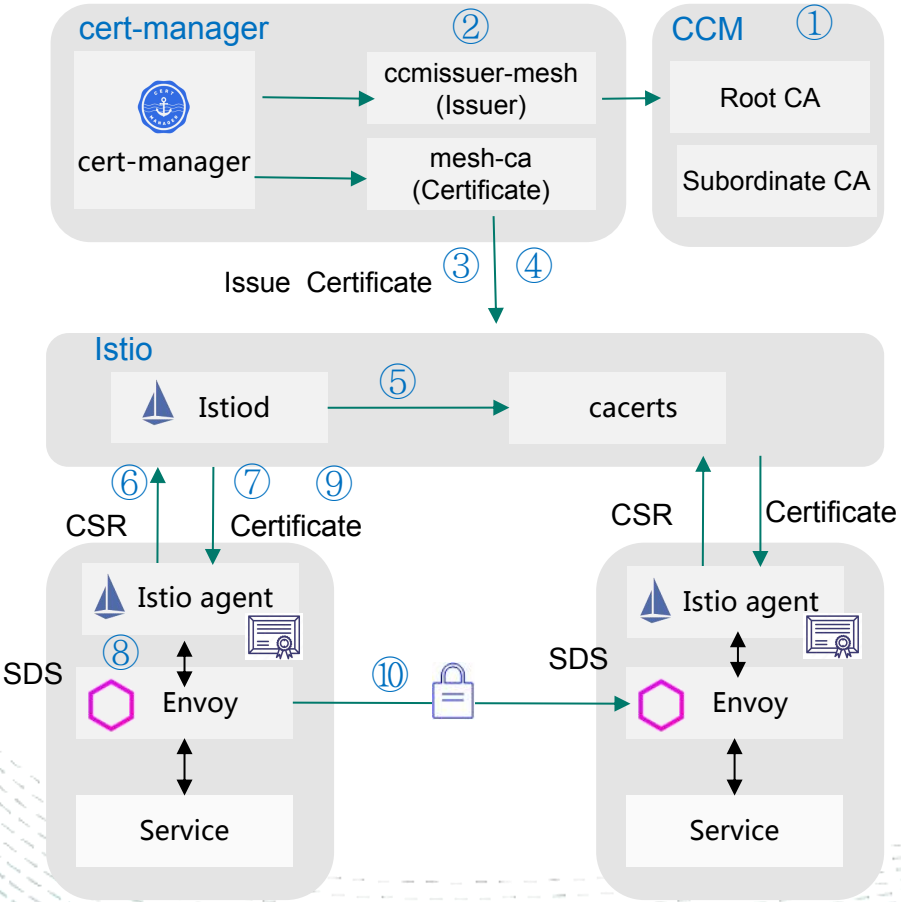
Issuer : are Kubernetes resources that represent certificate authorities (CAs) that are able to generate signed certificates by honoring certificate signing requests

Certificate: Certificates define a desired X.509 certificate which will be renewed and kept up to date. Certificates resources allow you to specify the details of the certificate you want to request. They reference an issuer to define how they'll be issued.

Agenda

- Background: Istio Zero Trust Security
- Challenge: Certificate Management In Istio
- Solution: cert-manager
- Practice: cert-manager Issues Certificate For Mesh Root CA
- Practice: cert-manager Issues Certificate For Ingress Gateway

Architecture



① Create PCA CA hierarchy, include root CA and a subordinate CA

② Configure to make subordinate CA as cert-manager issuer

③ cert-manager issues certificate as root CA, and store in secret cacerts, according to certificate mesh-ca configuration.

④ cert-manager check and renew certificates at a configured time in mesh-ca before expiry.

⑤ istiod offers a gRPC service to take certificate signing requests (CSRs)

⑥ Istio agent creates the private key and CSR, and sends CSR to istiod for signing

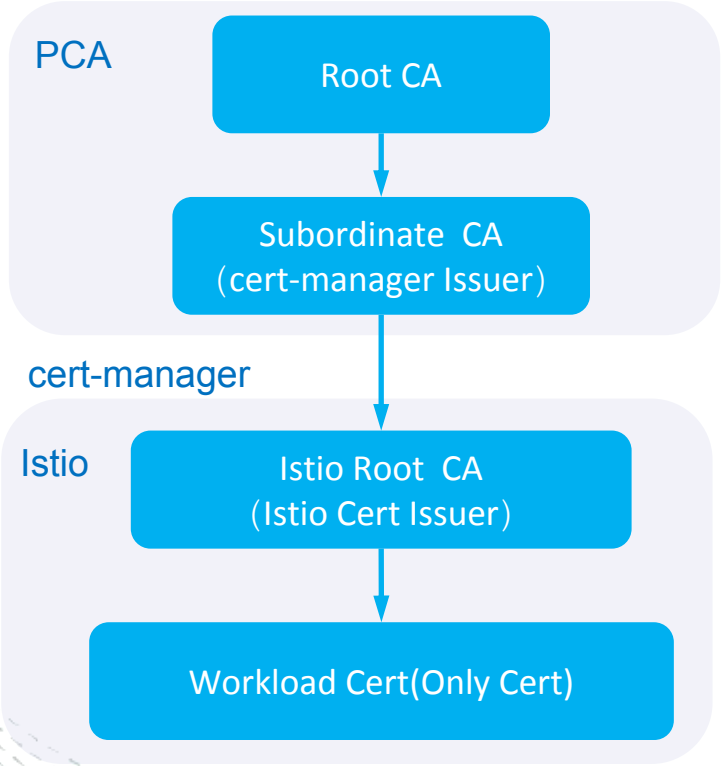
⑦ Istiod signs the CSR to generate the certificate

⑧ Envoy requests the certificate and key from the Istio via Envoy secret discovery service (SDS) API

⑨ Istio agent monitors the expiration of the workload certificate, periodically check for certificate and key rotation

⑩ The proxies of each workload communicate with other workloads by mutual TLS (mTLS) with workload certificate

CA Hierarchy (1)



Root Certificates

The public key certificate of a CA. A root certificate is the trust anchor in the public key infrastructure (PKI) system. It can issue subordinate CAs, private certificates, and certificate revocation lists (CRLs). After a root CA is imported into the client trust list, the certificates issued by it can be validated as trusted

Subordinate Certificates

Subordinate are CA certificates signed by another CA. Most intermediates will be signed by a root certificate, but it's possible to construct longer chains where an intermediate can be signed by another intermediate. Subordinate certificates are usually issued with a much shorter lifetime than the CA which signed them.

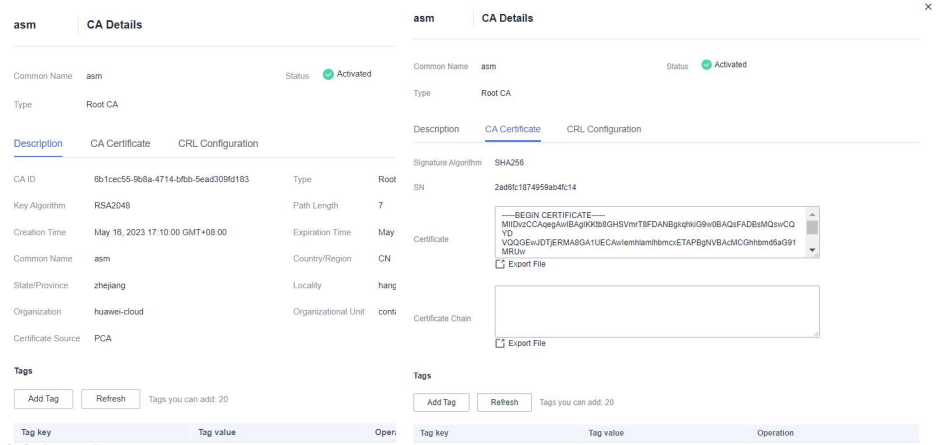
Leaf Certificates

Leaf certificates are usually used to represent a particular identity, rather than being used to sign other certificates. On the Internet leaf certificates usually identify a particular domain.

CA Hierarchy (2)

Common Name	Status	Type	Key Alg...	Organiz...	Issued CA	Expiration Time	Billing ...	Operation
asm-subordinate-ca 4bd1b320-e162-419f-8add-dfcc046197c7	Activated	Subordinate...	RSA2048	container	asm	Apr 27, 2024 11:35:34 GMT+08:00	Yearly/Monthly 31 days unti...	Export CA C
asm 6b1cec55-9b8a-4714-bfbb-5ead309fd183	Activated	Root CA	RSA2048	container	--	May 16, 2024 17:09:59 GMT+08:00	Pay-per-Use Created on ...	Export CA C
asm-subordinate 5fa7296b-aad3-4d85-b52d-0a70732c0988	Expired	Subordinate...	RSA2048	container-1	asm	Jun 16, 2023 17:12:22 GMT+08:00	Pay-per-Use Created on ...	Export CA C

CA List



#IstioCon

CA

Subordinate CA

Configuration: cert-manager Certificate

```
apiVersion: ccm-issuer.ccm.com/v1beta1
kind: CCMIssuer
metadata:
  labels:
    app.kubernetes.io/name: ccmissuer
    app.kubernetes.io/created-by: ccm-issuer
  name: ccmissuer-mesh
spec:
  urn: "4bd1b320-e162-419f-8add-dfcc046197c7"
  region: "cn-north-4"
  secretRef:
    …
    namespace: istio-system
    name: ccm-secret
```

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: mesh-ca
  namespace: istio-system
spec:
  isCA: true
  duration: 7200h #300d
  renewBefore: 240h #10d
  secretName: cacerts
  commonName: mesh-ca
  subject:
    organizations:
    - cluster.local
  issuerRef:
    group: ccm-issuer.ccm.com
    kind: CCMIssuer
    name: ccmissuer-mesh
```

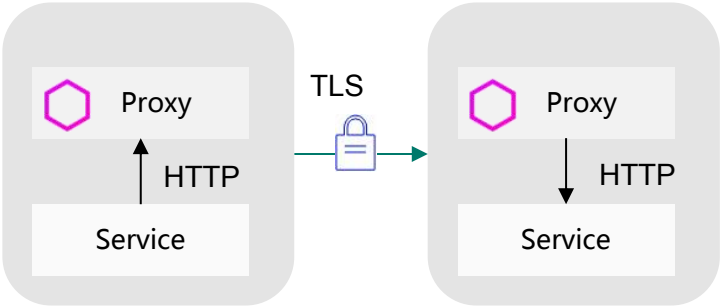

Configuration: Istio mTLS

```
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: nginx
  namespace: accesslog
spec:
  mtls:
    mode: STRICT
  selector:
    matchLabels:
      app: nginx
```

```
apiVersion: networking.istio.io/v1beta1
kind: DestinationRule
metadata:
  name: nginx
  namespace: accesslog
spec:
  host: nginx
  subsets:
  - labels:
      version: v1
    name: v1
    trafficPolicy:
      tls:
        mode: ISTIO_MUTUAL
```


Test: Transparent mTLS

```
# curl -v -s nginx:80
* Rebuilt URL to: nginx:80/
*   Trying 10.246.91.131...
* TCP_NODELAY set
* Connected to nginx (10.246.91.131) port 80 (#0)
> GET / HTTP/1.1
> Host: nginx
> User-Agent: curl/7.52.1
> Accept: */*
>
< HTTP/1.1 200 OK
< server: envoy
< date: Wed, 06 Sep 2023 06:09:04 GMT
< content-type: text/html
< content-length: 615
< last-modified: Tue, 28 Mar 2023 15:01:54 GMT
< etag: "64230162-267"
< accept-ranges: bytes
< x-envoy-upstream-service-time: 2
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```



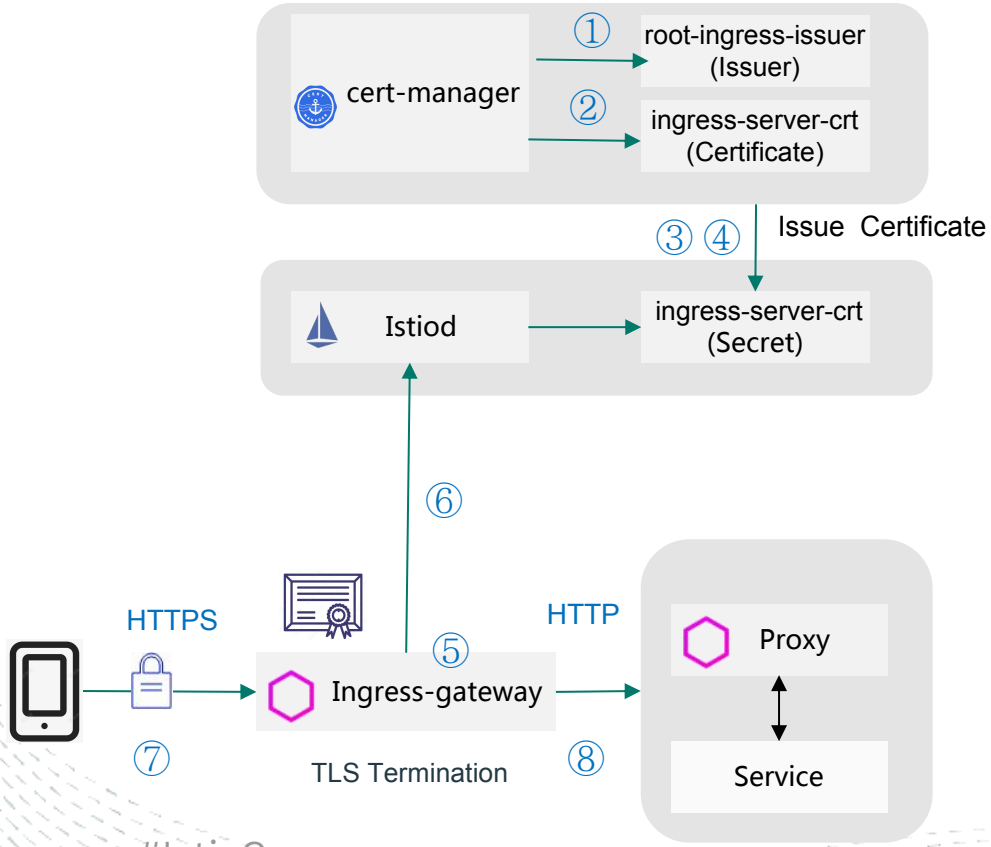
[2023-09-06T06:09:04.949Z] "GET / HTTP/1.1" 200 - via_upstream - "-" 0 615 2 2 "-" "curl/7.52.1" "0c25359b-a6f7-46c0-bb25-d39194c33205" "nginx" "10.66.1.2:80" outbound|80|v1|nginx.accesslog.svc.cluster.local 10.66.0.39:37200 10.246.91.131:80 10.66.0.39:52026 - -

[2023-09-06T06:09:04.951Z] "GET / HTTP/1.1" 200 - via_upstream - "-" 0 615 0 0 "-" "curl/7.52.1" "0c25359b-a6f7-46c0-bb25-d39194c33205" "nginx" "10.66.1.2:80" inbound|80|| 127.0.0.6:53389 10.66.1.2:80 10.66.0.39:37200 outbound_.80_.v1_.nginx.accesslog.svc.cluster.local default

Agenda

- Background: Istio Zero Trust Security
- Challenge: Certificate Management In Istio
- Solution: cert-manager
- Practice: cert-manager Issues Certificate For Mesh Root CA
- Practice: cert-manager Issues Certificate For Ingress Gateway

Architecture



① Create Issuer root-ingress-issuer in cert-manager

② Create certificate ingress-server-crt in cert-manager

③ Issuer issue certificate and store in configured secret ingress-server-crt

④ cert-manager check and renew certificates at a configured time in mesh-ca before expiry

⑤ Gateway TLS reference certificate secret

⑥ Istiod dynamically push certificate to Ingress-gateway

⑦ Client make HTTPS request with server CA, handshaking with Ingress-gateway, verify certificate

⑧ Ingress-gateway make TLS termination, and forward HTTP request to backend service.

Configuration: cert-manager Certificate

```
apiVersion: cert-manager.io/v1
kind: Issuer
metadata:
  name: root-ingress-issuer
  namespace: istio-system
spec:
  ca:
    secretName: root-ingress-certs
```

ca type issuer

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ingress-server-crt
  namespace: istio-system
spec:
  isCA: false
  duration: 7200h #300d
  renewBefore: 240h #10d
  secretName: ingress-server-crt
  commonName: ingress-server-crt
  dnsNames:
  - cert-test.com
  issuerRef:
    name: root-ingress-issuer
    kind: Issuer
    group: cert-manager.io
```

Configuration: Gateway Route

```
apiVersion: networking.istio.io/v1beta1
kind: Gateway
metadata:
  name: cert-gw-https
  namespace: istio-system
spec:
  selector:
    istio: ingressgateway
  servers:
  - hosts:
    - cert-test.com
    port:
      name: https
      number: 1028
      protocol: https
    tls:
      credentialName: ingress-server-crt   # cert
generated by cert-manager
      mode: SIMPLE
```

```
apiVersion: networking.istio.io/v1beta1
kind: VirtualService
metadata:
  name: cert-gw-https-962a97e6
  namespace: istio-system
spec:
  gateways:
  - istio-system/cert-gw-https
  hosts:
  - cert-test.com
  http:
  - match:
    - uri:
        prefix: /
    route:
    - destination:
        host: tomcat.cert-test.svc.cluster.local
```


Parse: Certificate Content

kubectl get certificate

```
[root@cluster-zcm-66173-v1o7e coded]# kubectl get  certificat
e  ingress-server-crt  -nistio-system -oyaml
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  creationTimestamp: "2023-09-02T12:58:02Z"
  generation: 2
  name: ingress-server-crt
  namespace: istio-system
  resourceVersion: "19291650"
  uid: e9b48e13-d331-40b7-bacf-04dedf23249b
spec:
  commonName: ingress-server-crt
  dnsNames:
  - cert-test.com
  duration: 7200h0m0s
  issuerRef:
    group: cert-manager.io
    kind: Issuer
    name: root-ingress-issuer
  renewBefore: 240h0m0s
  secretName: ingress-server-crt
status:
  conditions:
  - lastTransitionTime: "2023-09-06T07:28:34Z"
    message: Certificate is up to date and has not expired
    observedGeneration: 2
    reason: Ready
    status: "True"
    type: Ready
  notAfter: "2024-07-02T07:28:34Z"
  notBefore: "2023-09-06T07:28:34Z"
  renewalTime: "2024-06-22T07:28:34Z"
  revision: 2
```

config_dump secret

```
    {
     "name": "kubernetes://ingress-server-crt",
     "version_info": "2023-09-06T10:28:05+08:00/
     "last_updated": "2023-09-06T07:28:34.596Z",
     "secret": {
      "@type": "type.googleapis.com/envoy.extens
      "name": "kubernetes://ingress-server-crt",
      "tls_certificate": {
       "certificate_chain": {
        "inline_bytes": "LS0tLS1CRUdJTiBDRVJUSUZ
kcKTVNjd0VBWURWUVFLRXdsdFpYTm9MbU5sY25Rd0V3WURWU
Zd056STRNelJhRncweU5EQTNNREl3Ck56STRNelJhTUIweEd
W9DZ2dFQkFMWklWWEZ4SzZLTXl5YUFqOGRXY1VFVkVoNTNlN
FaTVtMjQrRVVHcTJ4RmhNay9mQkxjeHlnbVFqcDdvRmZoR3F
mdKTk5wb1IvRzhiWjNCV3NzcgpYL25SZTdNdzNVNGViZ0VrU
PdzJhUUtaY3hjNWJpaFVJVituQkdnMmpMa0NBd0VBQWFOYk1
XliWGtXSzQ1VHRmZVc4d0dBWURWUjBSQkJFd0Q0SU5ZMlZ5Z
UTDZMR0ZCT2U0R1NYZVkKaElNU280Q3owUUtVdFBlSXcvL0o
zFxakxaQTFFb2FmTk13REp0bTdWQnFmbW9iSW93ekVWOVVXC
hcXlEZkFWN0pWM3ppbExLQWFOMkxPVmtkeUJhS3pTQU1aUUE
lRJRklDQVRFLS0tLS0KLS0tLS1CRUdJTiBDRVJUSUZJQ0FUR
Fbk1CQUdBMVVFQ2hNSmJXVnphQzVqWlhKME1CTUdBMVVFQ2h
1hEVEkyTURJeU1qQTNNamN3Ck1Gb3dSakVuTUJBR0ExVUVDa
NdFkyVnlkSE13Z2dFaU1BMEdDU3FHU0liM0RRRUJBUVVBQTR
Xp6QnhlK3k2RDBGUnZJd3FPdkZNNURUeWFRMzZGWWJxR1NYR
2UHJKYlR1Tyt5cWNCcgpXbGwwKzI0bU1uOS9NUi9sWW1OdUh
lViOWdScGF0TlVRcTduVEdtemk5bVdKYmlueWdFQW1lWisKV
nTlZIUk1CQWY4RUJUQURBUUgvTUIwR0ExVWREZ1FXQkJRazQ
2txaGtpRzl3MEIKQVFzRkFBT0NBUUVBWnhxY040WG9GOEt2e
zN2lQUmJqaGNDajZkTUZuWUw2bEp6dnNuWFJNdnhSClMrUEY
3VNcUJ1QTRrNnJDL1FVYVRBQzRaQ1ZIa3RMU3VaaU55T01sd
wQXZNRmxKCkdTZk1kSkRCbjBUanB4OU9Lc1dIMU43cnZxVWF
       },
       "private_key": {
        "inline_bytes": "W3JlZGFjdGVkXQ=="
```

| base64 -d | openssl x509 -noout -text

```
VJUSUZJQ0FURS0tLS0tCg== | base64 -d | openssl x509 -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            58:3b:d2:6f:ee:f4:2d:d6:c7:f4:f1:14:3d:1a:99:69
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: O = mesh.cert + O = cert-manager, CN = root-ingress-certs
        Validity
            Not Before: Sep  6 07:28:34 2023 GMT
            Not After : Jul  2 07:28:34 2024 GMT
        Subject: CN = ingress-server-crt
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b6:48:55:71:71:2b:a2:8c:cb:26:80:8f:c7:56:
                    71:41:15:12:1e:77:7b:ac:97:f7:d9:34:c3:3b:ed:
                    9f:b1:a4:94:40:e1:4a:f7:c4:45:53:d4:ed:4d:ed:
                    e9:5c:2c:89:25:0c:e3:3b:cf:f9:77:1e:c8:a8:86:
                    f8:fc:76:2a:cf:c8:4c:19:67:ce:7d:19:53:2d:d1:
                    44:48:b9:9b:6e:3e:11:41:aa:db:11:61:32:4f:df:
                    04:b7:31:ca:09:90:8e:9e:e8:15:f8:46:a9:df:b6:
                    2d:6a:89:59:b4:43:82:6d:03:a4:78:28:69:bf:cc:
                    99:67:ca:09:7f:d0:2b:f7:11:f8:b9:39:af:b1:ff:
                    a5:c6:f6:f9:36:82:4b:27:07:8a:99:91:a6:15:5b:
                    0a:b7:23:ac:a3:87:46:38:09:34:da:68:47:f1:bc:
                    6d:9d:c1:5a:cb:2b:5f:f9:d1:7b:b3:30:dd:4e:1e:
                    6e:01:24:44:71:df:fe:28:a6:80:77:c0:12:2e:42:
                    55:be:01:92:fc:e4:25:db:bb:10:3f:d0:6b:39:4a:
                    48:66:50:98:f1:72:d5:32:d7:17:01:c2:da:4a:a0:
                    61:d0:95:fd:da:86:9e:ed:7f:2f:ed:c4:c0:ec:36:
                    69:02:99:73:17:39:6e:28:54:21:5f:a7:04:68:36:
                    8c:b9
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Authority Key Identifier:
                keyid:24:E3:8F:5F:DD:BC:34:72:49:23:26:D7:91:62:B8:E5:3B:5F:79:6F

            X509v3 Subject Alternative Name:
                DNS:cert-test.com
    Signature Algorithm: sha256WithRSAEncryption
         af:e9:d8:71:59:d3:5d:29:40:d8:c9:54:1c:62:08:34:db:9e:
         41:e2:28:f5:c4:cc:69:32:fa:2c:61:41:39:ee:06:49:77:98:
         84:83:12:a3:80:b3:d1:02:94:b4:f7:88:c3:ff:c9:d1:b3:bc:
         f3:f4:79:5d:30:34:35:0c:e4:3a:ef:d3:47:68:5a:0c:e8:f8:
         4e:d3:9b:da:ef:87:5a:08:dc:d1:c0:39:54:90:99:f0:36:69:
         28:62:9b:00:11:c4:c1:91:d6:16:ee:4d:2a:73:4b:93:5a:a3:
         2d:90:35:12:86:9f:34:cc:03:26:d9:bb:54:1a:9f:9a:86:c8:
         a3:0c:c4:57:d5:16:4d:c6:27:d3:44:02:63:25:34:d7:3f:b9:
         c0:77:b8:58:5d:e2:73:58:50:e2:80:1d:8f:04:75:f4:99:0c:
         06:f9:1b:3e:b7:37:7e:99:55:8d:ea:27:93:3e:d3:d2:18:0c:
         9a:f1:d5:4d:3d:b6:6a:ac:83:7c:05:7b:25:5d:f3:8a:52:ca:
         01:a3:76:2c:e5:64:77:20:5a:2b:34:80:31:94:00:e7:68:f3:
         1b:9e:12:e4:66:43:cf:16:a0:34:88:9a:aa:3d:d8:8b:eb:50:
         3f:3c:42:4d:a3:68:e6:e7:ab:ab:21:84:60:d6:c9:cc:ad:95:
         b7:8d:9c:47
```

Test: Ingress TLS Termination

```
[root@cluster-zcm-66173-v1o7e coded]#  curl --cacert ingress-server-cert-
ca.crt -v -HHost:cert-test.com --resolve cert-test.com:443:100.85.115.86
 https://cert-test.com:443/
* Added cert-test.com:443:100.85.115.86 to DNS cache
* Hostname cert-test.com was found in DNS cache
*   Trying 100.85.115.86:443...
* Connected to cert-test.com (100.85.115.86) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*  CAfile: ingress-server-cert-ca.crt
*  CApath: none
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=ingress-server-crt
*  start date: Sep  6 07:28:34 2023 GMT
*  expire date: Jul  2 07:28:34 2024 GMT
*  subjectAltName: host "cert-test.com" matched cert's "cert-test.com"
*  issuer: O=mesh.cert + O=cert-manager; CN=root-ingress-certs
*  SSL certificate verify ok.
* Using HTTP2, server supports multiplexing
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade
: len=0
* Using Stream ID: 1 (easy handle 0x562fcc8ec6d0)
> GET / HTTP/2
> Host:cert-test.com
> user-agent: curl/7.79.1
> accept: */*
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Connection state changed (MAX_CONCURRENT_STREAMS == 2147483647)!
< HTTP/2 200
< content-type: text/html;charset=UTF-8
< date: Wed, 06 Sep 2023 08:11:14 GMT
< x-envoy-upstream-service-time: 3
< server: istio-envoy
```

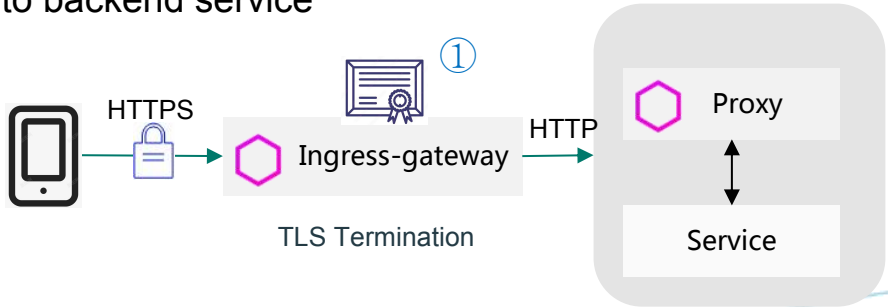
① Ingress-gateway load certificate which configured by cert-manager

② Client make HTTPS request with server CA

③ TLS handshake between client and Ingress-gateway

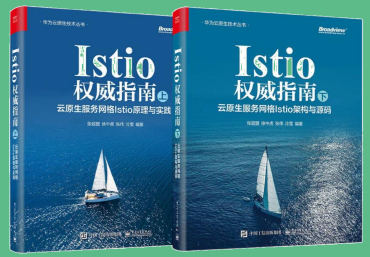
④ Verify certificate, include expire date， subjectAltName

⑤ Ingress-gateway offload TLS and make a HTTP request to backend service



⑤ [2023-09-06T06:42:45.383Z] "GET / HTTP/2" 200 - via_upstream - "-" 0 11230 4 3 "172.16.0.96" "curl/7.79.1" "78c0ce88-dc95-495c-adf2-be59584448e2" "cert-test.com" "10.66.1.12:8080" outbound|8080||tomcat.cert-test.svc.cluster.local 10.66.0.9:48024 10.66.0.9:1028 172.16.0.96:10021 cert-test.com tomcat.cert-test.svc.cluster.local:8080

Thank you!



#IstioCon