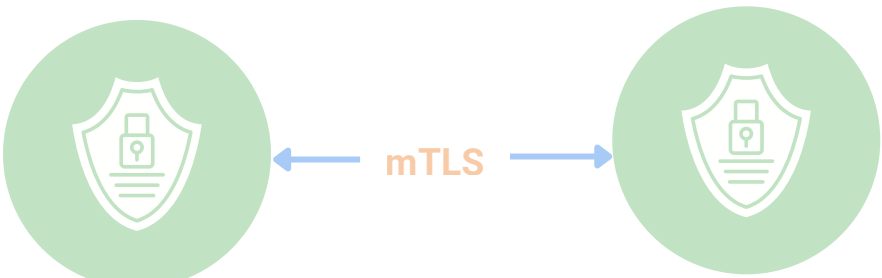
MTLS WITH ISTIO SERVICE MESH

Table of Contents

What is mTLS

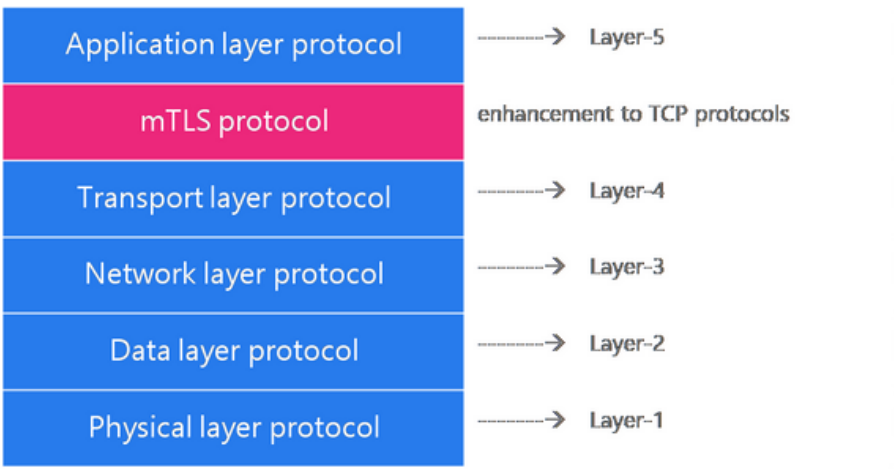
Mutual Transport Layer Security (mTLS) is a cryptographic protocol designed to authenticate two parties and secure their communication in the network. mTLS protocol is an extension of TLS protocol where both the parties- web client and web server- are authenticated. The primary aim of mTLS is to achieve the following:

- Authenticity: To ensure both parties are authentic and verified
- Confidentiality: To secure the data in the transmission
- Integrity: To ensure the correctness of the data being sent



mTLS protocol: A part of TCP/IP suite

mTLS protocol sits between the application and transport layers to encrypt only messages (or packets). It can be seen as an enhancement to the TCP protocol. The below diagram conceptually provides the location of mTLS in the TCP/IP protocol suite.



The idea is that all the messages (or packets) from the application layer will be encrypted using the mTLS protocol and will be sent to the TCP layer for transmission to the receiver.

03

SSL vs TLS vs mTLS: Which is new?

Security engineers, architects and developers use SSL, TLS, and mTLS interchangeably, often because of their similarity. Loosely mentioning, mTLS is an enhancement to TLS, and TLS is an enhancement to SSL.

The first version of Secure Socket Layer (SSL) was developed by Netscape corporate in 1994; the most popular versions were versions 2 and 3- created in 1995. It was so widely popular among banks that it made its way into the script of many Hollywood movies in the 1990s.

The overall working of SSL is carried by three sub-protocol:

- Handshake protocol: used to authenticate the web client and the web server and establish a secured communication channel. In the handshaking process, a shared key will be generated, for the session only, to encrypt the data during communication.
- Record protocol: helps to maintain the confidentiality of data in the communication between the client and the server using a newly generated shared secret key.
- Alert protocol: In case the client or the server detects an error, the alert protocol would close the SSL connection (the transmission of data will be terminated); destroying all the session, shared keys, etc.

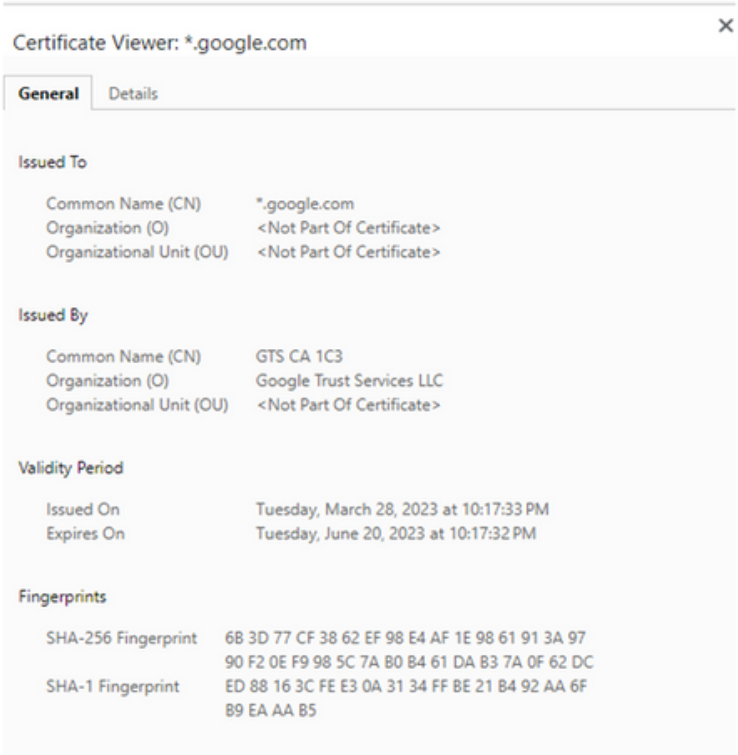
SSL vs TLS vs mTLS: Which is new?

As there were more internet applications, the requirement for fine-grain security of the data in the network was more. So Transport Layer Security (TLS) - a standard internet version of SSL- was developed by IETF. Netscape handed over the SSL project to IETF, and TLS is an advanced version of SSL; the code idea and implementation of the protocol are the same.

The main difference between the SSL and TLS protocols is that the cipher suite (or the algorithms) used to encrypt data in TLS is advanced. Secondly, the handshake, record, and alert protocols are modified and optimized for internet usage.

Note: In the SSL handshake protocol, the server authentication to the client by sending the certificate was mandatory, but the client's authentication was optional to secure the line. But in TLS, there was only a provision to authenticate we-servers to the client, not vice-versa. Almost all the websites you visit with HTTPS as the protocol will use TLS certificates to establish themselves as genuine sites. If you visit Google.com and click the padlock symbol, it will show the TLS certificates.

SSL vs TLS vs mTLS: Which is new?



The TLS was mainly used for web applications with the client being the user. Additionally, ensuring the authentication of billions of clients or users is only feasible for some web applications.

But as the large monolithic applications broke into numerous microservices that communicate over the internet, the need for mTLS grew suddenly. mTLS protocol ensures both the web client and the web server authenticate themselves before a handshake. (We will see the working model of the mTLS protocol later in this article).

Why mTLS is important than ever?

Modern business is done using web applications whose underlying architecture follows a hybrid cloud model. Microservices will be distributed across public/private clouds, Kubernetes, and on-prem VMs. And the communication among various microservices and components happens over the network, posing a significant risk of hacking or malicious attacks. Below are a few scenarios of cyber-attacks on the web that can be avoided entirely by using mTLS protocols.

Man-in-the-middle attack (MITM): Attackers can place themselves between a client and a server to intercept the data during the transmission. When mTLS is used, attackers cannot authenticate themselves and will fail to steal the data.

IP Spoofing: Another case is when bad guys masquerade as someone you trust and injects malicious packets into the receiver. This is again solved by end-point authentication in mTLS to determine with certainty if network packets or the data originates from a source we trust.

Packet Sniffer: The attacker can place a passive receiver near the wireless transmitter to obtain a copy of every packet transmitted. Such an attack is prevalent in Banking and Fintech domains when an attacker wants to steal sensitive information such as card numbers, banking application usernames, passwords, SSNs, etc. Since packet sniffing is non-intrusive, it is tough to detect. Hence the best way to protect data is to involve cryptography. mTLS helps encrypt the data using complex cryptographic algorithms that are hard to decipher by packet sniffers.

Why mTLS is important than ever?

PDenial-of-service (DoS) attacks: The attackers aim to make the network or the web server unusable by legitimate applications or users. This is done by sending vulnerable packets, or deluge to packets, or by opening a large number of TCP connections to the hosts (or the web server) so that the server ultimately crashes. DoS and Distributed DoS (advanced DoS technique) can be avoided by invoking mTLS protocols in the applicable communication. All the malicious DoS attacks will be discarded before entering into the handshake phase.

Use cases of mTLS in the industry

The use cases of mTLS are growing daily with the increasing usage of business through web applications and the simultaneous rise in threats of cyber attacks. Here are a few important use cases based on our experiences while discussing with many leaders from various industries or domains- banking, fintech, and online retail companies.

- **Hybrid cloud and multicloud applications:** Whenever every organization uses a mix of data centers- on-prem and public/private cloud- then the data leaves the secured perimeter, and goes out of the network. In such cases, mTLS should be used to protect the data.

- **Microservices-based B2B software:** Many B2B software in the market follows a microservices architecture. Each service would talk to the other using REST APIs. Even though all the services are hosted in a single data center, the network should be secured to protect the data in transit (in case the firewall is breached).

- **Online retail and e-commerce application:** Usually, e-commerce and online retail applications use Content Delivery Network (CDN) to fetch the application from the server and show it to users. Although TLS is implemented in the CDN to authenticate itself when a user visits the page, there should be a security mechanism to secure the network between the CDN and the web server through mTLS.

Use cases of mTLS in the industry

- **Banking applications:** Applications that carry susceptible transactions, such as banks, financial transaction apps, payment gateways, etc., should take extreme precautions to prevent their data from getting stolen. Millions of online transactions happen every day using various banking and fintech apps. Sensitive information such as bank usernames, passwords, debit/credit card details, CVV numbers, etc., can be easily hacked if the data in the network is not protected. Strict authentication and confidentiality can be applied to the network using mTLS.

- **Industry regulation and compliance:** Every country will have some rules and standards to govern the IT infrastructure and protect the data. All the policies, such as FIPS, GDPR, PCI-DSS, HIPAA, ISO27001, etc., outline strict security measures to protect the data-at-rest and data-in-transit. For strict authentication in the network, mTLS can be used, and companies can adhere to various standards.

In the next section we will discuss a few concepts one needs to be aware of before understanding the mechanism of how mTLS works.

Certificate, Public/Private Keys: Must know concepts about mTLS

Certificates

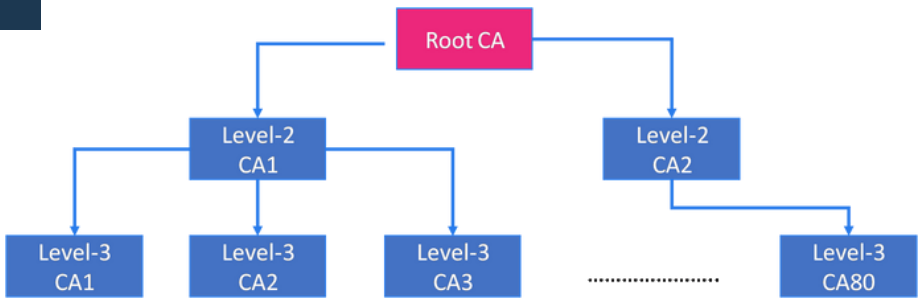
A (digital) certificate is a small computer file issued by a certificate authority (CA) to authenticate a user, an application, or an organization. A digital certificate contains information such as- the name of the certificate holder, serial number of the certificate, expiry date, public key, and signature of the certificate issuing authority.

Certificate Authority (CA)

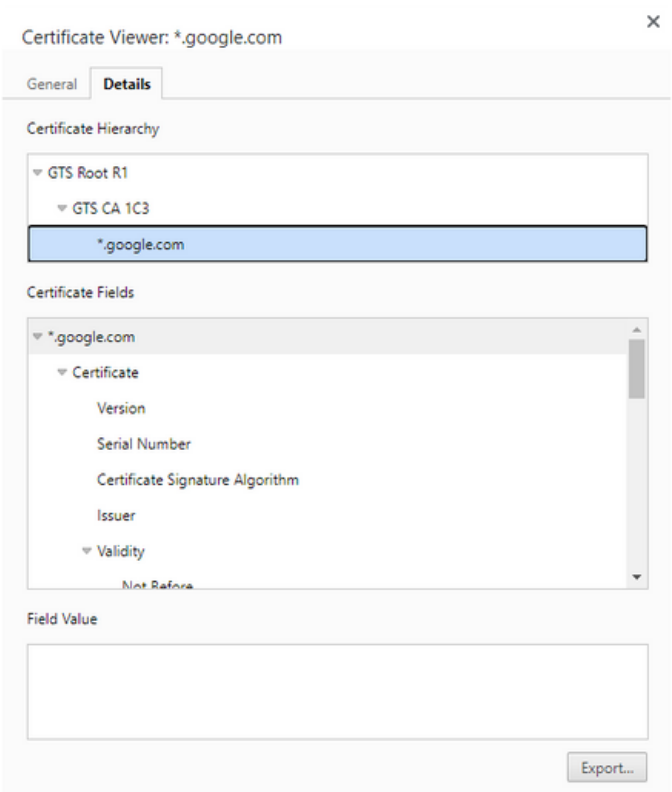
A certificate authority (CA) is a trusted 3rd party that verifies user identity and issues an encrypted digital certificate containing the applicant's public key and other information. Notable CAs are VeriSign, Entrust, LetsEncrypt, Safescript Limited, etc.

Root CA/ Certificate Chain

Certificate Authority hierarchies are created to distribute the workloads of issuing certificates. There can be entities issuing certificates from different CA at various levels. In the multi-level hierarchy (like parent and child) of CAs, there is one CA at the top, called the Root CA (refer the below image). Each CA would also have its certificate issued by the parent CA, and the root CA will have self-signed certificates.



To ensure the CA (which issued the certificate to the client/server) is trusted, the security protocol suggests that entities send their digital certificate and the entire chain leading up to the root CA.



Public and Private Key Pair

While creating certificates for an entity, the CA would generate a public and a private key- commonly called a public key pair. The public and private keys are used to authenticate their identity and encrypt data. Public keys are published, but the private key is kept secret. If you are interested to learn about the algorithms to generate public keys- RSA, DSA, ECDSA, ed25519).

X.509 Certificate

It is a special category of the certificate, defined by the International Telecommunications Union, which binds an application's identity (hostname, organization name, etc.) to a public key using a digital signature. It is the most commonly used certificate in all the security protocols SSL/TLS/mTLS for securing web applications.

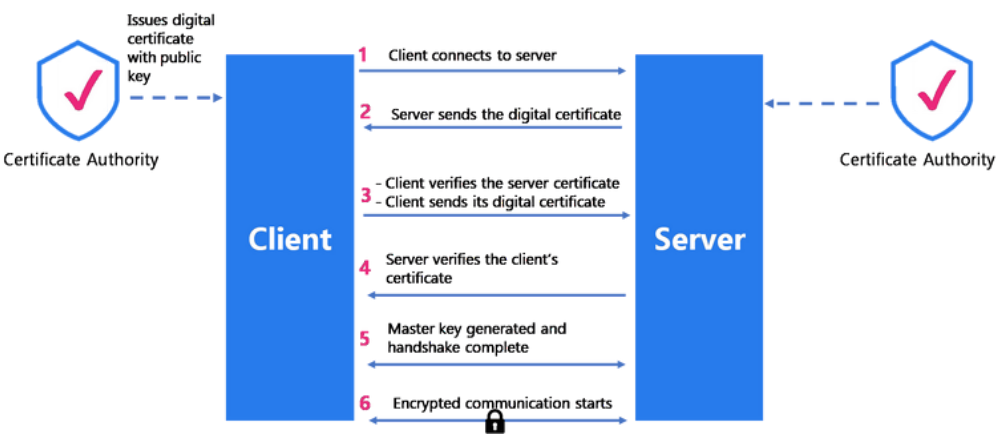
How does mTLS work

As explained earlier, the mTLS has a similar implementation of sub-protocols as SSL. There are 9 phases (mentioned below) for two applications to talk to each other using the mTLS protocol.

1. **Establish security capabilities with hello:** The client tries to communicate with the server (also known as client hello). The client hello message would contain values for specific parameters such as mTLS version, session id, Cipher suite, compression algorithm, etc. The server also would send a similar response called server hello with the values (it supports) for the same parameters sent by the client.
2. **Server authentication and key exchange:** In this phase, the server would share its digital certificate (mostly X.509 certificates for microservices) and the entire chain leading up to root CA to the client. It would also request the client's digital certificate.
3. **Client verifies the server's certificate:** The client would use the public key in the digital certificate to validate the server's authenticity.
4. **Client authentication and key exchange:** After validation, the client sends a digital certificate to the server for verification.
5. **Server verifies client's certificate:** The server verifies the client's authenticity.
6. **Master key generation and handshake complete:** Once the parties' authenticity is established, the client and server will establish a handshake, and two new keys will be generated; shared secret information is only known to the parties and active for the session.
    - Master secret: for encryption
    - Message Authentication Code (MAC): for assuring message integrity

How does mTLS work

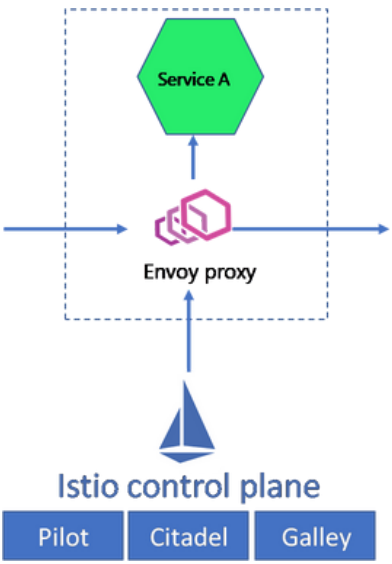
7. **Communication encrypted and transmission starts:** The exchange of the information will begin with all the messages or packets encrypted using the master secret key. Behind the veil, the mTLS protocol will divide the message into smaller blocks called fragments, compress each fragment, add the MAC for each block, and finally encrypt them using the master secret.
8. **Data transmission starts:** Finally, the mTLS protocol will append headers to the blocks of messages and send it to TCP protocol to send it to the destination or receiver.
9. **Session ends:** Once the communication completes, the session will close. If an anomaly is detected during the transmission, the mTLS protocol will destroy all the keys and secrets and terminate the session immediately.



Note, in the above phases, we have assumed that the CA would have issued a certificate to the entities which are still valid. In reality, the certificate of mission-critical applications expires soon, and there is a requirement for constant certificate rotation (we will straight away jump into how Istio enables mTLS and certificate rotation).

How to enable mTLS and certificate rotation using Istio service mesh

Istio service mesh is an infrastructure layer that abstracts out the network and security later out of application layers. It does so by injecting an Envoy proxy (an L4 and L7 sidecar proxy) into each application and listening to all the network communication.



mTLS implementation in Istio

Though Istio supports multiple authentication types, it is best known for implementing mTLS to applications hosted over the cloud, on-prem, or Kubernetes infrastructure. The Envoy proxy acts as Policy Enforcement Points (PEP); you can implement mTLS using the peer-to-peer (p2p) authentication policy provided by Istio and enforce it through the proxies at the workload level.

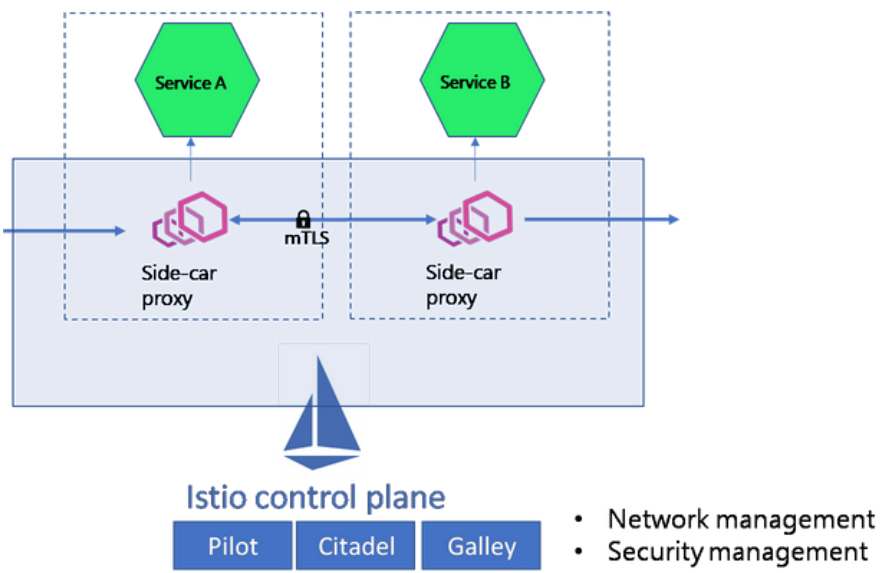
Example of p2p authentication policy in Istio to apply mTLS to **demobank** app in the 'istio-nm' namespace :

```
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: "mTLS-peer-policy"
  namespace: "istio-nm"
spec:
  selector:
    matchLabels:
      app: demobank
  mtls:
    mode: STRICT
```

mTLS implementation in Istio

The working mechanism of mTLS authentication in Istio is as follows:

1. At first, all the outbound and inbound traffic to any application in the mesh is re-routed through the Envoy proxy.
2. So the mTLS happens between the client-side Envoy proxy and the server-side Envoy proxy.
3. The client-side Envoy proxy would try to connect with the server-side Envoy proxy by exchanging certificates and proving their identity.
4. Once the authentication phase is completed successfully, a TCP connection between the client and service side Envoy proxy is established to carry out encrypted communications.



Note the mTLS with Istio can be implemented at all levels- application, namespace, or mesh-wide.

Certificate management and rotation in Istio service mesh

Istio provides stronger identity by issuing X.509 certificates to Envoy proxies attached to applications. The certificate management and rotation is done by an Istio agent running in the same container as Envoy proxy. The Istio agents talk to the Istiod- the control plane of Istio- to effectively circulate the digital certificates with public keys. Below are the details phases of certificate management in Istio:

1. Istio agents generate public key pairs (private and public keys) and then send the public key to the Istio control plane for signing. This is called a certificate signing request (CSR).
2. Istiod has a component (earlier Galley) that acts as the CA. Istiod validates the public key in the request, signs, and issues a digital certificate to the Istio agent.
3. When mTLS connection is required, Envoy proxies fetch the certificate from the Istio agent using Envoy secret discovery service (SDS) API.
4. The Istio agent observes the expiration of the certificate used by the Envoy. Upon the certificate's expiry, the agent initiates a CSR to Istiod.

Certificate management and rotation in Istio service mesh

Istio provides stronger identity by issuing X.509 certificates to Envoy proxies attached to applications. The certificate management and rotation is done by an Istio agent running in the same container as Envoy proxy. The Istio agents talk to the Istiod- the control plane of Istio- to effectively circulate the digital certificates with public keys. Below are the details phases of certificate management in Istio:

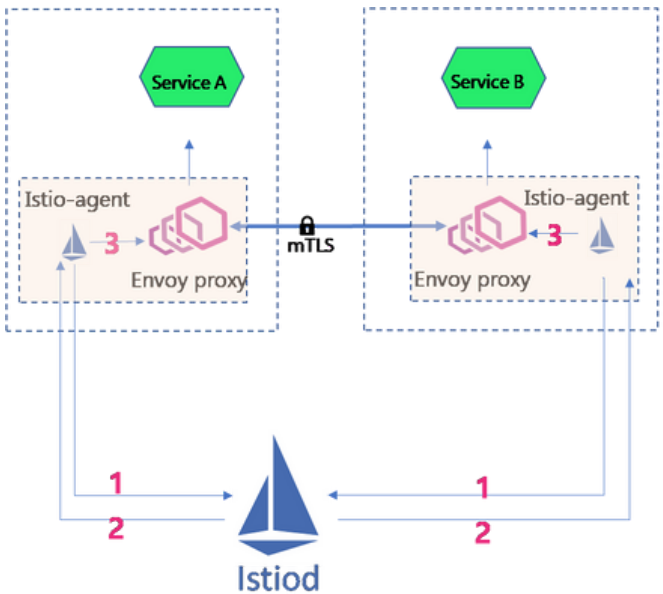
1. Istio agents generate public key pairs (private and public keys) and then send the public key to the Istio control plane for signing. This is called a certificate signing request (CSR).

2. Istiod has a component (earlier Galley) that acts as the CA. Istiod validates the public key in the request, signs, and issues a digital certificate to the Istio agent.

3. When mTLS connection is required, Envoy proxies fetch the certificate from the Istio agent using Envoy secret discovery service (SDS) API.

4. The Istio agent observes the expiration of the certificate used by the Envoy. Upon the certificate's expiry, the agent initiates a CSR to Istiod.

Certificate management and rotation in Istio service mesh



Certificate Management Steps in Istio

1. Istio agent initiates certificate signing request
2. Istio validiate, signs and issue the certificate
3. Envoy proxy fetch the requests from the agent for establishing mTLS

Next Step

If you want to implement Istio, please check our Youtube video on securing multicloud and multicluster applications using mTLS and L7 authorization with Istio. There is even a blog if you plan to check out the implementation.

To help you overcome the toil of learning and experimenting with Istio, IMESH provides enterprise Istio support. With IMESH, you can implement Istio for your application spread across cloud, containers or on-prem VMs from Day-1 without any operational hassle. We provide:

- Istio implementation into production and securing of multicloud apps
- Enhancements like multi-cluster implementation in AKE/GKE/EKS, FIPS compliance
- Training and onboarding of Istio
- Istio lifecycle management with frequent patches and version upgrades
- High-performance and Highly available Istio
- Guaranteed SLAs for vulnerability fixes

Authors

Ravi Verma, CTO, IMESH

Ravi, a technology visionary, brings 12+ years of experience in software development and cloud architecture in enterprise software. He has led R&D divisions at Samsung and GE Healthcare and architected high-performance, secure and scalable systems for Baxter and Aricent. His passion and interest lie in network and security. Ravi frequently discusses open-source technologies such as Kubernetes, Istio, and Envoy Proxy from the CNCF landscape.

Debasree Panda, CEO, IMESH

Debasree understands customer pain points in cloud and microservice architecture. Previously, he led product marketing and market research teams at Digitate and OpsMx, where he had created a multi-million dollar sales pipeline. He has helped open-source solution providers- Tetrate, OtterTune, and Devtron- design GTM from scratch and achieve product-led growth. He firmly believes serendipity happens to diligent and righteous people.

About IMESH

IMESH offers Kubernetes-native application network and security platform to manage multi-cloud and hybrid cloud environments. The IMESH platform is built on top of Istio service mesh and Envoy API gateway and helps cloud, platform and security teams to make Kubernetes application more secure, manageable, and reliable.

Visit: https://imesh.ai/
email: contact@imesh.ai