AWS ALB mTLS Device Certificate POC — Step-by-Step SOP

Purpose: Restrict access to an ERP application so that only authorized Windows devices with a valid client certificate can establish HTTPS access through an AWS Application Load Balancer (ALB).

POC URL: `https://apps.ditos.io`

ALB: `hello-world-alb`
HTTPS Listener: `443`
Target Group: `hello-world-tg`
AWS Region: `ap-south-1`

> **Important:** This SOP records the actual POC journey, including an initial approach that did not work with Chrome and the corrected approach. Do not skip the correction notes.

---

1. Target Architecture

```
Authorized Windows Laptop
        |
        | HTTPS + Client Certificate
        | TPM-backed private key
        v
AWS Application Load Balancer
        |
        | mTLS verification
        | Trust Store -> Ditos ERP POC Root CA
        v
ERP Target Group
        |
        v
ERP Application
        |
        | Normal ERP username/password
        v
Authenticated User
```

Security layers

1. **TLS server authentication** — `apps.ditos.io` server certificate.
2. **Device authentication** — AWS ALB verifies the Windows client certificate.
3. **User authentication** — ERP still requires normal username/password.
4. The client private key is generated using **Microsoft Platform Crypto Provider** and is **not exportable**.

2. Prerequisites

Mac / CA administration machine

- macOS
- OpenSSL 3.x
- Terminal access
- Secure storage for the CA private key

Windows authorized device

- Windows 10/11
- TPM 2.0 available and ready
- Administrator PowerShell access
- Chrome or Edge
- Device name: `DITOS-PC-001`

AWS

- Existing Application Load Balancer
- Existing HTTPS listener on port 443
- Existing server certificate for `apps.ditos.io`
- Permission to create S3 bucket, trust store and modify ALB listener

---

3. Create the POC Root CA on Mac

Create the working directory:

```
mkdir -p ~/Documents/SSLBlockTest/{ca,client,config}
cd ~/Documents/SSLBlockTest
```

Verify OpenSSL:

```
openssl version
```

Example:

```
OpenSSL 3.6.4
```

3.1 Generate Root CA private key

The Root CA private key must remain on the secure CA/admin machine.

Example:

```
openssl genrsa -aes256 -out ca/ditos-erp-root-ca.key 4096
```

Protect the passphrase.

Never upload the Root CA private key to AWS S3 or to the Windows client.

3.2 Create Root CA certificate

Example:

```
openssl req -x509 -new -sha256 \
  -key ca/ditos-erp-root-ca.key \
  -out ca/ditos-erp-root-ca.crt \
  -days 3650 \
  -subj "/C=IN/ST=Karnataka/L=Bengaluru/O=Ditos/OU=IT/CN=Ditos ERP POC Root CA"
```

Verify:

```
openssl x509 -in ca/ditos-erp-root-ca.crt -noout -subject -issuer -dates
```

The Root CA should show:

```
CA:TRUE
```

---

4. Create Client Certificate Extensions

Create:

```
~/Documents/SSLBlockTest/config/client-ext.cnf
```

Contents:

```
basicConstraints=critical,CA:FALSE
keyUsage=critical,digitalSignature
extendedKeyUsage=clientAuth
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
```

The important setting is:

```
extendedKeyUsage=clientAuth
```

This makes the certificate suitable for TLS client authentication.

---

5. Verify Windows TPM

On `DITOS-PC-001`, open **Administrator PowerShell**.

Run:

```
Get-Tpm
```

Expected important values:

```
TpmPresent : True
TpmReady   : True
TpmEnabled : True
```

If TPM is not present/ready, stop and resolve the TPM issue before continuing.

---

6. Initial Client Certificate Approach — IMPORTANT CORRECTION

6.1 Initial approach

The first certificate was generated using a Windows machine key with:

```
MachineKeySet = TRUE
ProviderName = "Microsoft Platform Crypto Provider"
Exportable = FALSE
```

This created a certificate for:

```
CN=DITOS-PC-001
```

and it was successfully signed by the POC Root CA.

The certificate was installed in:

```
Cert:\LocalMachine\My
```

and verification showed:

```
Provider = Microsoft Platform Crypto Provider
Private key is NOT exportable
HasPrivateKey = True
```

Why this was not sufficient

Although the certificate was correctly TPM-backed, Chrome did not use this certificate for the browser client-authentication flow.

Chrome showed a different client certificate and the request to:

```
https://apps.ditos.io
```

resulted in:

```
ERR_CONNECTION_RESET
```

Evidence from testing

Chrome's certificate viewer showed a certificate with:

```
CN=4a1a0212-5e58-4891-8950-bdce3040dfe7
```

instead of:

```
CN=DITOS-PC-001
```

Therefore, **do not assume that a machine-store certificate being TPM-backed automatically means Chrome will select it for browser mTLS.**

---

7. Corrected Approach — User Certificate Store + TPM

For the browser POC, the corrected approach was to generate the TPM-backed private key in the **current user's certificate context**.

The private key remains:

- TPM-backed
- non-exportable
- managed by Microsoft Platform Crypto Provider

7.1 Create Windows CSR configuration

On Windows:

```
@"
[Version]
Signature="$Windows NT$"

[NewRequest]
Subject = "CN=DITOS-PC-001, O=Ditos, OU=ERP Devices, L=Bengaluru, ST=Karnataka, C=IN"
KeyLength = 2048
KeyAlgorithm = RSA
HashAlgorithm = SHA256
KeyUsage = 0x80
ProviderName = "Microsoft Platform Crypto Provider"
ProviderType = 0
RequestType = PKCS10
Exportable = FALSE

[Extensions]
2.5.29.37 = "{text}1.3.6.1.5.5.7.3.2"
"@ | Out-File -Encoding ascii C:\DitosMTLS\PC-001-user.inf
```

Generate CSR:

```
certreq -new C:\DitosMTLS\PC-001-user.inf C:\DitosMTLS\PC-001-user.csr
```

Verify:

```
certutil -dump C:\DitosMTLS\PC-001-user.csr
```

Confirm:

```
Subject: CN=DITOS-PC-001
Public Key Length: 2048 bits
Enhanced Key Usage:
    Client Authentication (1.3.6.1.5.5.7.3.2)
Provider = Microsoft Platform Crypto Provider
```

---

8. Sign Windows CSR on Mac

Copy:

```
C:\DitosMTLS\PC-001-user.csr
```

to:

```
~/Documents/SSLBlockTest/client/PC-001-user.csr
```

On Mac:

```
cd ~/Documents/SSLBlockTest
```

Sign:

```
openssl x509 -req \
  -in client/PC-001-user.csr \
  -CA ca/ditos-erp-root-ca.crt \
  -CAkey ca/ditos-erp-root-ca.key \
  -CAcreateserial \
  -out client/PC-001-user.crt \
  -days 365 \
  -sha256 \
  -extfile config/client-ext.cnf
```

Verify:

```
openssl verify \
  -CAfile ca/ditos-erp-root-ca.crt \
  client/PC-001-user.crt
```

Expected:

```
client/PC-001-user.crt: OK
```

---

9. Install Signed Certificate on Windows

Copy only:

```
PC-001-user.crt
```

to:

```
C:\DitosMTLS\PC-001-user.crt
```

Do not copy a private key.

The private key must remain on the Windows device/TPM.

Install:

```
certreq -accept C:\DitosMTLS\PC-001-user.crt
```

Record the returned thumbprint.

In the completed POC the thumbprint was:

```
0D87688BF26E15388D1A720E9C48F339EF0D0947
```

---

10. Verify Certificate Store

Check the current user's certificate store:

```
Get-ChildItem Cert:\CurrentUser\My |
Where-Object {$_.Thumbprint -eq "0D87688BF26E15388D1A720E9C48F339EF0D0947"} |
Format-List Subject,Thumbprint,HasPrivateKey
```

Expected:

```
Subject       : CN=DITOS-PC-001, ...
Thumbprint    : 0D87688BF26E15388D1A720E9C48F339EF0D0947
HasPrivateKey : True
```

Check the machine store:

```
Get-ChildItem Cert:\LocalMachine\My |
Where-Object {$_.Thumbprint -eq "0D87688BF26E15388D1A720E9C48F339EF0D0947"} |
Format-List Subject,Thumbprint,HasPrivateKey
```

For this corrected browser POC, the certificate should be in:

```
Cert:\CurrentUser\My
```

---

11. Verify TPM and Non-Exportable Private Key

Run:

```
certutil -user -store My 0D87688BF26E15388D1A720E9C48F339EF0D0947
```

The completed POC showed:

```
Provider = Microsoft Platform Crypto Provider
Private key is NOT exportable
Encryption test passed
```

These are critical checks.

Required security properties

Property	Expected
Certificate subject	`CN=DITOS-PC-001`
Provider	Microsoft Platform Crypto Provider
Private key	Present
Exportable	No
Key algorithm	RSA
Key length	2048
EKU	Client Authentication
Certificate issuer	Ditos ERP POC Root CA

---

12. Configure AWS S3 for Trust Store

Create a private S3 bucket, for example:

```
ditos-erp-mtls-poc-9835
```

Upload only:

```
ditos-erp-root-ca.crt
```

Do **not** upload:

```
ditos-erp-root-ca.key
```

Keep S3 public access blocked.

---

13. Create AWS ALB Trust Store

In AWS:

```
EC2
→ Load Balancers
→ Trust Stores
→ Create trust store
```

Name:

```
ditos-erp-poc-trust-store
```

Use the Root CA certificate bundle containing:

```
ditos-erp-root-ca.crt
```

Expected status:

```
Active
```

The completed POC showed:

```
CA certificates in bundle: 1
Total revoked certs: 0
```

---

14. Enable mTLS on ALB HTTPS Listener

Navigate:

```
EC2
→ Load Balancers
→ hello-world-alb
→ HTTPS:443 listener
→ Security
→ Edit secure listener settings
```

Configure:

Client certificate handling

```
Mutual authentication (mTLS): ON
```

Mode

```
Verify with trust store
```

Trust store

```
ditos-erp-poc-trust-store
```

Expired client certificate handling

Select:

```
Don't allow expired client certificates
```

Advertise trust store CA subject names

For the POC:

```
ON
```

Do not change

Keep the existing server certificate:

```
apps.ditos.io
```

Do not modify the HTTP listener `80` unless specifically required.

Save the listener configuration.

---

15. Verify AWS Listener After Saving

The ALB listener security page should show:

```
Mutual authentication (mTLS)
Verify with trust store
```

Trust store:

```
ditos-erp-poc-trust-store
```

Trust store association status:

```
Active
```

Advanced settings:

```
Expired client certificates:
Don't allow expired client certificates

Advertise trust store CA subject names:
On
```

---

16. Browser Certificate Verification

On the authorized Windows laptop, open Chrome:

```
chrome://certificate-manager/clientcertificates
```

Under:

```
Your certificates
```

Chrome should be able to access certificates imported from Windows.

The corrected certificate should correspond to:

```
DITOS-PC-001
```

If Chrome displays a completely different certificate, investigate before continuing.

---

17. Authorized Device Test

On `DITOS-PC-001`:

```
https://apps.ditos.io
```

Chrome should request/select the client certificate.

Select:

```
DITOS-PC-001
```

After successful TLS/mTLS authentication, the ERP website should open.

If the ERP then asks for its normal username/password, that is expected.

Final authentication flow

```
Device certificate
        ↓
AWS ALB mTLS verification
        ↓
HTTPS connection accepted
        ↓
ERP application
        ↓
ERP username/password
        ↓
User authenticated
```

---

18. Unauthorized Device Test

Use a second Windows laptop that does **not** have the authorized client certificate/private key.

Open:

```
https://apps.ditos.io
```

Do not install the client certificate.

Expected result:

```
Access denied / TLS client certificate error / connection reset
```

The exact browser error can vary.

The important requirement is:

> The unauthorized laptop must not establish a successful HTTPS session with the ERP through the ALB.

---

19. Security Validation

The POC is considered successful when all of the following are true:

- [x] Authorized laptop has TPM.
- [x] Client private key generated using Microsoft Platform Crypto Provider.
- [x] Client private key is non-exportable.
- [x] Client certificate has Client Authentication EKU.
- [x] Client certificate is signed by the trusted Root CA.
- [x] Root CA certificate is in the ALB trust store.
- [x] ALB HTTPS listener uses mTLS Verify mode.
- [x] Trust store association is Active.
- [x] Authorized laptop can access `apps.ditos.io`.
- [ ] Unauthorized laptop without client certificate is blocked.
- [ ] ERP username/password remains required after mTLS.

---

20. Important Lessons / Corrections From the POC

Correction 1 — Machine certificate vs browser certificate store

Initial attempt

Certificate was installed in:

```
Cert:\LocalMachine\My
```

It was correctly TPM-backed:

```
Provider = Microsoft Platform Crypto Provider
Private key is NOT exportable
```

However, Chrome did not use that certificate for the browser client-authentication flow.

Corrected approach

The client certificate was generated with the TPM-backed key and installed in:

```
Cert:\CurrentUser\My
```

with:

```
HasPrivateKey = True
```

Chrome could then select the appropriate client certificate.

Correction 2 — Never copy the private key to the client

The Mac CA signs the CSR but does not receive the Windows private key.

The correct flow is:

```
Windows TPM
   |
   | generates private key
   |
   +--> CSR
          |
          v
       Mac CA
          |
          | signs CSR
          v
       Client CRT
          |
          v
       Windows
          |
          +--> associates certificate with existing TPM key
```

Never do:

```
Windows private key → Mac
```

and never do:

```
CA private key → Windows
```

Correction 3 — Do not use a Mac-generated client private key for the final Windows POC

An earlier Mac-generated client key/certificate existed:

```
DITOS-PC-001.key
DITOS-PC-001.csr
DITOS-PC-001.crt
```

These were not used for the final TPM-backed Windows browser authentication.

The final path uses:

```
PC-001-user.csr
PC-001-user.crt
```

with the private key generated on Windows using:

```
Microsoft Platform Crypto Provider
```

---

21. Troubleshooting

Browser shows ERR_CONNECTION_RESET

First check:

1. Is mTLS enabled on the ALB?
2. Is the trust store association Active?
3. Is the client certificate trusted by the ALB trust store?
4. Does Chrome see the correct client certificate?
5. Does the certificate have Client Authentication EKU?
6. Does the certificate have an associated private key?
7. Is the private key accessible through Microsoft Platform Crypto Provider?

Do not immediately change the ALB configuration.

Chrome shows an unexpected certificate

If Chrome shows a certificate with an unrelated CN, for example:

```
CN=4a1a0212-5e58-4891-8950-bdce3040dfe7
```

instead of:

```
CN=DITOS-PC-001
```

inspect:

```
chrome://certificate-manager/clientcertificates
```

and verify the Windows certificate store.

Certificate has HasPrivateKey = False

The certificate cannot perform client authentication.

Check:

```
Get-ChildItem Cert:\CurrentUser\My |
Where-Object {$_.Thumbprint -eq "<THUMBPRINT>"} |
Format-List Subject,Thumbprint,HasPrivateKey
```

Expected:

```
HasPrivateKey : True
```

Certificate is in LocalMachine but Chrome does not use it

This was encountered during the POC.

For the browser POC, use the corrected current-user TPM certificate approach and verify:

```
Cert:\CurrentUser\My
```

---

22. Operational Certificate Lifecycle

The client certificate in this POC was issued for:

```
365 days
```

Before expiration:

1. Generate a new CSR on the authorized Windows device.
2. Generate the new private key through TPM.
3. Sign the CSR using the CA.
4. Install the new certificate.
5. Verify the new certificate has the TPM-backed private key.
6. Test access.

7. Revoke/retire the old certificate if the PKI supports revocation.
8. Remove old certificates after successful migration.

---

23. Production Recommendations

This POC uses a locally managed Root CA. For production, consider:

- AWS Private CA or an enterprise PKI.
- HSM/protected CA private keys.
- Automated certificate issuance and renewal.
- Certificate revocation strategy / CRLs.
- Device inventory and certificate-to-device mapping.
- Microsoft Intune / enterprise device management.
- Endpoint compliance controls.
- Monitoring and ALB access logging.
- Certificate expiry alerts.
- Formal certificate rotation process.

Important limitation

A TPM prevents extraction/export of the private key material, but it does not by itself guarantee that a compromised authorized Windows machine cannot use the private key. An attacker with sufficient control of the authorized endpoint may potentially cause the machine to perform cryptographic operations with the key.

For stronger production security, combine mTLS with endpoint management, device compliance and identity controls.

---

24. Final POC Checklist

Area	Status
Root CA created	Complete
Root CA private key protected	Complete
Windows TPM verified	Complete
Initial machine-store approach tested	Failed for Chrome browser selection
Corrected user-store TPM approach	Complete
Client CSR generated on Windows	Complete
Client CSR signed by CA	Complete
Client certificate installed	Complete
Client certificate has private key	Complete
Microsoft Platform Crypto Provider	Complete
Private key non-exportable	Complete

Area	Status
AWS S3 CA bundle	Complete
AWS ALB trust store	Complete
ALB mTLS Verify mode	Complete
Trust store association	Active
Authorized laptop access	**Working**
Unauthorized laptop blocked	**To be tested**
ERP username/password	**To be confirmed after mTLS**

---

25. POC Conclusion

The corrected implementation successfully demonstrates:

```
TPM-backed device identity
        +
AWS ALB mTLS
        +
ERP application authentication
```

The authorized Windows laptop can access:

```
https://apps.ditos.io
```

while the next validation step is to confirm that a second laptop without the authorized client certificate is blocked.

This SOP should be used as the POC baseline, while the production implementation should replace the locally managed CA process with an enterprise-grade PKI/certificate lifecycle.