Secure and Centralized Secret Handling with mTLS in Azure-Certificate-Secret-Proxy

This project is a community-built reference implementation for secure secret delivery to managed endpoints (Windows, macOS, Linux) using mutual TLS (mTLS).

GitHub repository: https://github.com/lucanoahcaprez/Azure-Certificate-Secret-Proxy

TL;DR

This solution enables certificate-based, centralized secret retrieval for managed endpoints without distributing static secrets to clients. Devices authenticate with mTLS, the function enforces validation policy, and secrets are fetched on demand from controlled backends.

Problem Statement

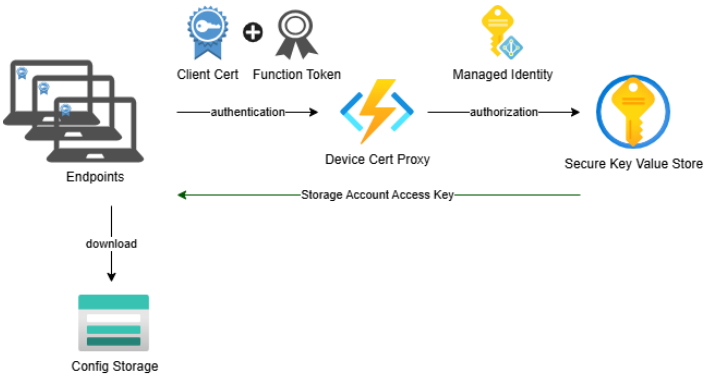
The core issue is persistent secret exposure on endpoints due to packaging and script distribution patterns.

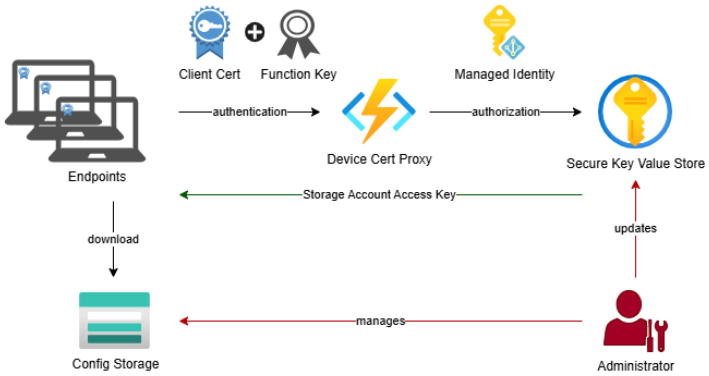
Many endpoint automation workflows still expose secrets in clear text or leave them recoverable on disk (for example in remediation scripts, package payloads, or local logs). This creates long-lived secret exposure on clients.

This solution removes static secret distribution and shifts secret access to a certificate-authenticated, server-side retrieval model.

Technical Overview

This is the general architecture:





The request path is intentionally minimal and policy-driven so trust and retrieval decisions stay server-side.

1. Client presents a device certificate during TLS handshake.
2. Azure App Service enforces client-certificate requirement.
3. Function receives and validates certificate (`X-ARR-ClientCert`).
4. Validation policy is applied (`EntraDeviceCert`, `CertChainValidation`, `TrustedThumbprints`, or combination).
5. Requested secret is retrieved from selected backend (`APPSETTINGS`, `KEYVAULT`, `TABLE`).
6. Response is returned as scoped JSON payload.

Security Properties

The following controls define the baseline security posture of the current implementation.

Control	Implementation
Strong client auth	mTLS with device certificate proof-of-possession

Control	Implementation
Policy-based trust	Entra device binding, custom root trust, thumbprint allowlist
Secret minimization	Per-request retrieval, no static local secret bundle
Credentialless backend auth	Managed Identity for Azure APIs
Operational diagnostics	Structured response diagnostics and centralized logging

Additional Notes

The following sections cover practical application scope, fit, and roadmap direction.

Example Use Cases

Typical scenarios are operational tasks that currently rely on static or embedded secrets.

- Shared device local admin password retrieval
- License/key distribution (for example ESU)
- BIOS/firmware secret retrieval
- Storage SAS URL delivery
- Log ingestion bootstrap (for example LAW/DCR scenarios)
- Central configuration values for certificate-enabled endpoints
- Server scenarios where IMDS/Arc onboarding is not available

Architecture Fit

This implementation is a reusable pattern, not a one-size-fits-all product.

This is a concept architecture and should be adapted to tenant-specific controls, compliance boundaries, and operational requirements.

Planned V2 Enhancements

Planned improvements focus on cryptographic hardening, resilience, and operational visibility.

- Application-layer payload encryption with ephemeral key exchange
- Automated key rotation primitives
- Rate limiting and abuse controls (plus DDoS hardening)
- Network isolation with private endpoints

- Improved telemetry, traceability, and anomaly detection

Documentation Entry Points

Use the links below for implementation details, deployment instructions, and architecture deep dives.

- https://github.com/lucanoahcaprez/Azure-Certificate-Secret-Proxy/tree/main/docs
- https://github.com/lucanoahcaprez/Azure-Certificate-Secret-Proxy/blob/main/docs/ARCHITECTURE.md
- https://github.com/lucanoahcaprez/Azure-Certificate-Secret-Proxy/blob/main/docs/DEPLOY.md

---

Revision #2
Created 2026-05-27 13:27:30 UTC by Luca Noah Caprez
Updated 2026-05-27 13:29:40 UTC by Luca Noah Caprez