

From End-to-End Test Deployment to Ecosystem Coordination: Lessons from Deploying Merkle Tree Certificates

Affiliation: TrustAsia Technologies, Inc.

Abstract

This paper reports on TrustAsia's experience building an MTC test environment based on [draft-ietf-plants-merkle-tree-certs-05](#) [1]. Within the validation test realm of the Chrome Quantum-resistant Root Program (CQRP), we completed an end-to-end deployment covering ACME ordering and domain control validation, Issuance Log submission, cosigning by an independent external Mirror, issuance of both Standalone and Landmark-relative certificates, ARI-based automated renewal, dynamic TLS credential selection, and validation in Chrome Canary. The existing MTC, ACME, and log/Mirror specifications were sufficient for four independently developed implementations - an ACME client, an external Mirror, a TLS stack, and Chrome - to interop end to end. What the deployment actually consumed was calendar time: waiting for each component's release cycle to line up. We recommend that the Workshop promote a multi-implementation testing framework spanning all of these components, while treating non-Web-PKI use cases as a secondary topic for open discussion.

Keywords: Merkle Tree Certificates; ACME; TLS; Mirror; interoperability; post-quantum authentication

1. TrustAsia's End-to-End Deployment in the Test Realm

TrustAsia operates two test MTC CAs, which issue certificates with maximum lifetimes of 7 and 47 days, respectively, together with an Issuance Log for each CA. The test environment also includes a TrustAsia-operated Mirror, a public ACME service, and an MTC demonstration site (<https://mtcdemo.trustasia.cn/>). Both test CAs and the Mirror are listed in the CQRP Root Store [3] under the `UNTRUSTED_VALIDATION_ONLY` realm. Geomys has added both TrustAsia Issuance Logs to its `mirror.log-list` [4] and provides cosignatures for Standalone certificates and landmark checkpoints. Both MTC certificate types therefore have an independent cosignature validation path within the CQRP test realm.

The test ACME service covers order processing and domain control validation (DCV), followed by log submission, cosigning, and issuance of both certificate types by the CA. TrustAsia adapted the open-source `acme.sh` code for MTC [5], supporting certificate requests, asynchronous retrieval, deployment, and automatic renewal triggered within the ARI suggested window defined in RFC 9773 [6]. We also completed interoperability testing with CertLocker, an independent client implementation covering certificate requests, validation, and issuance.

On the server side, the demonstration site configures three credential types for the same identity: a conventional X.509 certificate, a Standalone Certificate, and a Landmark-relative Certificate. TrustAsia implemented dynamic selection based on the ClientHello `trust_anchors` extension, with fallback to X.509. On September 3, 2026, Chrome Canary `155.0.8039.0` (arm64), with the MTC test flags enabled and `cosigners.json` 2.0.6 loaded, successfully validated both a Standalone certificate carrying a Geomys cosignature and a Landmark-relative certificate relying on the corresponding cosigned checkpoint/landmark at <https://mtcdemo.trustasia.cn/>. These deployment results provide the basis for the findings below.

2. Deployment Findings

2.1 Adoption Is Not Synchronized Across Components

Our implementation suggests that the main obstacle to MTC deployment is not the absence of basic protocol mechanisms. MTC draft-05 defines the mechanisms needed for ACME retrieval, asynchronous alternate certificates, and TLS certificate selection. The CQRP Policy [7] and C2SP log specifications [8] further define operational requirements and interfaces among CAs, Issuance Logs, Mirrors, and cosigners. ACME clients, TLS libraries, and server software each implement their respective parts, and their delivery schedules will naturally differ.

ACME clients face two separate implementation tasks. The first is MTC certificate retrieval. A client must recognize the relevant certificate types and trust anchor information, accept the Standalone certificate when it first becomes available, and use `acme-optional-alternate` to handle the asynchronously generated Landmark-relative certificate. When the optional resource returns HTTP 202 with `Retry-After`, the client should retry that resource independently rather than block deployment of the

certificate that is already available. The second task is renewal scheduling. TrustAsia's implementation uses ARI [6]: the ACME server supplies a suggested renewal window through `renewalInfo`, and the client uses that window to place a new order and complete deployment. ARI determines when renewal should occur; `acme-optional-alternate` determines how a client retrieves another certificate that becomes available later for the same order. These are distinct functions.

Results from the MTC-adapted `acme.sh` code and interoperability testing with CertLocker show that these retrieval mechanisms can be implemented. The continuously running ARI renewal path also shows that the lifecycle of short-lived certificates can be automated. At the same time, an ACME client does not become MTC-capable merely by complying with RFC 8555 [9]. Implementers should explicitly adopt the MTC extensions and integrate retrieval, renewal, deployment, and failure handling into a complete workflow. The fact that the TLS server, Geomys, and Chrome Canary have implemented the required capabilities at their respective layers does not mean that different projects will ship them in the same versions or on the same schedule. Whether the protocol works is no longer the open question. The open question is whether the pieces ship together: today an operator can have an MTC-ready CA and a browser that validates it, and still fail in the middle because the ACME client or TLS library lags a release behind.

2.2 TLS Stacks Need Reusable Multi-Credential Interfaces

During the transition in which X.509 and MTC coexist, a server must be able to configure multiple certificate types, select an MTC certificate based on the ClientHello `trust_anchors` extension, and fall back to X.509 [2]. TLS stacks therefore need reusable interfaces for loading and selecting multiple credentials, so that higher-layer servers such as Apache and Nginx do not each have to implement low-level handshake logic.

OpenSSL is the underlying TLS library for a large body of server software. OpenSSL issue #32231 proposes porting `SSL_CREDENTIAL` from BoringSSL as a prerequisite for supporting Trust Anchor Identifiers. The issue remains open and in progress, with no milestone assigned [10]. Extending this work to cover MTC requirements, including `trust_anchors` processing, loading and selecting among multiple certificate types, X.509 fallback, and stable interfaces for higher-layer software, would reduce the integration cost for servers such as Apache and Nginx and help different implementations converge on a consistent deployment model.

2.3 Trust-Data Distribution Is Part of the Test Surface

Browser validation depends on more than a correctly encoded certificate. The identities and operational status of CAs and Mirrors, as well as trust data such as landmarks, must be registered, distributed, and delivered to clients in time. TrustAsia's browser-side validation succeeded only after both test CAs, the TrustAsia-operated Mirror, and the Geomys Mirror had entered the relevant test configuration, Geomys had produced the cosignatures, and Chrome Canary had received updated CQRP data. Cross-operator registration, status propagation, and client updates should be tested alongside issuance, cosigning, and TLS certificate selection.

2.4 Non-Browser Environments Lack a General Mechanism for Landmark Synchronization

Validation of a Landmark-relative certificate depends on the verifier obtaining the relevant Landmark state in time. The current test environment synchronizes this state through online updates, and browsers can use their software and trust-data update channels to distribute it. Non-browser environments, such as server-to-server TLS clients, dedicated devices, and private deployments, often follow different update schedules. Some also operate with intermittent connectivity or entirely offline. The individual Internet-Draft [draft-gray-plants-mtc-deploy-use-cases-00](#) [11] proposes distribution approaches, but their interoperability, offline update behavior, rollback handling, and failure fallback still need to be tested.

Non-browser deployments must explicitly define the landmark source, update cadence, failure handling, and criteria for fallback to Standalone MTC or X.509. No reusable, cross-implementation deployment model currently exists. Developing one will require coordinated work among CAs, Mirrors, TLS implementers, device vendors, and platform providers.

3. Proposed Actions

First, we propose a multi-implementation testing framework spanning ACME clients, CAs, Issuance Logs, Mirrors, Landmark distribution, TLS stacks and servers, and browser verifiers. At a minimum, the framework should cover one complete cross-layer scenario:

[ACME ordering and asynchronous retrieval -> Mirror discovery and cosigning -> TLS credential selection and X.509 fallback -> Chrome validation](#)

Component-level testing does not fully cover these integration points. The proposed scenario should test cases such as an ACME client reacting to an updated ARI suggested window, or a server selecting credentials for a browser with stale landmark state. Its purpose is to identify failures at these boundaries. TrustAsia is willing to make its existing public ACME service and demonstration site available as entry points for community interoperability testing. These endpoints are provided for testing only and do not come with production service-level or permanent-availability commitments.

In addition to the browser validation path, the framework should test Landmark retrieval, caching, update failures, and fallback behavior for non-browser verifiers.

Second, Web PKI and TLS provide the first concrete deployment path for MTC, but they should not be the only measure of its value. [draft-gray-plants-mtc-deploy-use-cases-00](#) [11] has begun exploring use cases including private PKI, code signing, software supply chains, and constrained devices. Identifying the identity model, log/Mirror topology, lifecycle mechanisms, and verifier profile required by each use case would help determine where MTC is applicable beyond the Web PKI.

4. Conclusion

TrustAsia's deployment spans four independently implemented elements: an ACME client, an external Mirror, a TLS stack, and Chrome Canary. The complete MTC path is operational within the CQRP validation test realm. The next step is to align support across clients, TLS stacks, server software, browsers, and trust-data distribution systems and make the combined deployment reproducibly testable. The test-realm implementation provides a starting point; production deployment still depends on multi-implementation consistency testing and sustained ecosystem coordination.

References

1. [draft-ietf-plants-merkle-tree-certs-05](#):
<https://datatracker.ietf.org/doc/html/draft-ietf-plants-merkle-tree-certs-05>
2. [draft-ietf-tls-trust-anchor-ids-04](#):
<https://datatracker.ietf.org/doc/html/draft-ietf-tls-trust-anchor-ids-04>
3. Chrome Quantum-resistant Root Program Root Store:
<https://googlechrome.github.io/chromerootprogram/cqrp/root-store/>
4. Geomys Magnolia staging mirror.log-list:
<https://github.com/geomys/magnolia/blob/main/staging/mirror.log-list.txt>
5. TrustAsia MTC-adapted acme.sh branch:
<https://github.com/trustasia-com/acme.sh/tree/feat/mtc-certificate-support>
6. RFC 9773, ACME Renewal Information (ARI) Extension:
<https://www.rfc-editor.org/rfc/rfc9773.html>
7. CQRP Draft Policy v0.3.0:
<https://googlechrome.github.io/chromerootprogram/cqrp/draft-policy/>
8. C2SP transparency-log specifications: <https://github.com/C2SP/C2SP>
9. RFC 8555, Automatic Certificate Management Environment (ACME):
<https://www.rfc-editor.org/rfc/rfc8555.html>
10. OpenSSL issue #32231, "Port SSL_CREDENTIAL from BoringSSL":
<https://github.com/openssl/openssl/issues/32231>
11. [draft-gray-plants-mtc-deploy-use-cases-00](#):
<https://datatracker.ietf.org/doc/html/draft-gray-plants-mtc-deploy-use-cases-00>