Atlas: Enabling Cross-Vendor Authentication for IoT

Anonymous Authors

Abstract. Cloud-mediated IoT architectures fragment authentication across vendor silos and create latency and availability bottlenecks for cross-vendor device-to-device (D2D) interactions. We present Atlas, a framework that extends the Web public-key infrastructure to IoT by issuing X.509 certificates to devices via vendor-operated ACME clients and vendor-controlled DNS namespaces. Devices obtain globally verifiable identities without hardware changes and establish mutual TLS channels directly across administrative domains, decoupling runtime authentication from cloud reachability. We prototype Atlas on ESP32 and Raspberry Pi, integrate it with an MQTT-based IoT stack and a Atlas-aware cloud, and evaluate it in smart-home and smart-city workloads. Certificate provisioning completes in under 6 s per device, mTLS adds only about 17 ms of latency and modest CPU overhead, and Atlas-based applications sustain low, predictable latency compared to cloud-mediated baselines. Because many major vendors already rely on ACME-compatible CAs for their web services, Atlas is immediately deployable with minimal infrastructure changes.

1 Introduction

IoT ecosystems have grown rapidly, enabling programmable automation across smart homes, smart cities, and industrial environments. This programmability is largely achieved through cloud-based platforms such as IFTTT [30], which allow users to compose cross-vendor device interactions using automation recipes. To simplify device management, commercial IoT platforms have adopted vertically integrated architectures in which each vendor operates its own *IoT Cloud* to handle provisioning, telemetry, firmware updates, and communication. This model offers convenience and centralized control, performing adequately in single-tenant environments like smart homes where devices belong to a common administrative domain.

However, the cloud-first model introduces fundamental limitations. Cross-vendor communication requires multi-hop paths: each device must first communicate with its vendor's cloud, and a third-party service then mediates the interaction. This indirection introduces variable latency, expands the attack surface, and creates dependencies on external infrastructure. These limitations become untenable in large-scale, multi-stakeholder deployments such as smart cities, where strict quality-of-service requirements (low latency, high throughput, and continuous availability) must be maintained across independently administered device populations.

An intuitive solution is to enable direct device-to-device (D2D) communication, eliminating cloud dependency for routine interactions. However, the current

IoT ecosystem lacks a standardized mechanism for secure cross-vendor authentication, an essential prerequisite for safe D2D communication. Current mechanisms typically rely on bearer tokens (*e.g.,* OAuth) mediated by cloud services, which are fundamentally application-layer constructs. This leaves the transport layer unauthenticated between devices, forcing reliance on the cloud for security. This limitation is particularly acute in multi-tenant settings where independently administered devices (*e.g.,* municipal infrastructure, emergency services, commercial systems) must securely interoperate. In the absence of a standard, deployments resort to workarounds that compromise security. Third-party platforms like IFTTT rely on shared bearer tokens that act as permanent keys, introducing significant attack surface if intercepted. Conversely, local methods such as MAC-address filtering or unauthenticated mDNS discovery provide no cryptographic identity, leaving devices vulnerable to simple spoofing attacks.

Fragmentation of Trust. The root cause of this interoperability failure is *trust fragmentation*. Each vendor operates as an isolated trust domain with its own certificate authority (CA) or proprietary authentication mechanism. Devices can authenticate to their vendor's cloud, but no shared trust anchor exists across vendor boundaries. A simple solution, in which vendors issue self-signed certificates, fails because it merely replicates the fragmentation problem: devices from different vendors have no basis for mutual trust. Consortium-based approaches, such as the Matter alliance [31], attempt to address this by establishing a shared root store among member vendors. While effective for managed environments like smart homes, such alliances rely on pre-established trust relationships and manual commissioning processes (*e.g.,* scanning QR codes) that do not scale to ad-hoc, city-wide interactions. In decentralized, multi-principal deployments where municipal authorities, local businesses, and emergency services operate independently, the "intranet" model of alliances breaks down; an "internet" model of global trust is required.

Our Approach. We observe that the Web faced an analogous trust fragmentation problem two decades ago, and solved it through a federated PKI model anchored in globally trusted CAs. The ACME protocol [11], popularized by Let's Encrypt [26], further democratized this model by automating certificate issuance, renewal, and revocation. Today, ACME enables any domain owner to obtain trusted certificates at no cost, and the protocol's security properties have been formally verified [12].

We present Atlas, a framework that adapts this Web PKI model to IoT. Atlas assigns each device a globally unique identity encoded as a subdomain under its vendor's domain (*e.g.,* `<uuid>.devices.vendor.com`). The vendor's IoT cloud infrastructure acts as an ACME client, performing domain validation and obtaining X.509 certificates on behalf of devices. The resulting certificates are provisioned to devices during manufacturing or via over-the-air updates. At runtime, devices authenticate directly using mutual TLS (mTLS), without requiring cloud mediation. This design preserves vendor control over device identity while enabling cross-vendor authentication through the shared trust anchor of the Web PKI.

Threat Model. Atlas operates under the Dolev-Yao model: a network-capable attacker can eavesdrop, intercept, modify, inject, and replay messages, but lacks physical device access. Vendors are trusted within their namespace, consistent with the Web PKI model where domain owners assert identities in their zones [22, 35, 69]. Device attestation [52, 28, 70] is complementary but out of scope.

Contributions. This paper makes the following contributions:

- **IoT-specific PKI framework.** We design Atlas, extending the ACME protocol to IoT. Atlas defines a DNS-based device identity namespace under vendor-controlled domains, integrates with existing cloud infrastructure, and supports the full certificate lifecycle (issuance, renewal, revocation), establishing the Web PKI as a shared cross-vendor trust anchor.
- **End-to-end prototype.** We implement Atlas across the IoT stack: a vendor cloud component for certificate management and a lightweight device-side client for resource-constrained platforms. The implementation (∼2.5K LoC) integrates with standard IoT messaging protocols and is open-sourced [6].
- **Performance and scalability.** We evaluate Atlas on three hardware platforms (RPi4, RPi0W, ESP32) and large-scale ns-3 simulations. Atlas provisions certificates in under 6 s, adds only 17 ms of latency per session, and achieves two to three orders of magnitude lower latency than cloud-mediated alternatives, scaling to thousands of devices.
- **Deployment feasibility.** We show that 12 of the top 20 IoT vendors already use ACME-compatible CAs [1], enabling near-zero-effort adoption. We analyze Atlas under the Dolev-Yao model and discuss backward compatibility, revocation, and privacy, confirming deployability without new infrastructure.

2 Background

IoT Cloud-based Deployment. Modern IoT ecosystems predominantly adopt a cloud-first deployment model, driven by architectural simplicity, scalability, and centralized control [5, 18]. This centralized architecture simplifies development, accelerates feature rollout, and allows analytics integration without device-side changes. In practice, each vendor cloud also acts as the primary trust anchor for its devices, managing credentials and enforcing authentication policies. However, this model enforces vendor lock-in: all communication is routed through proprietary clouds, and local or cross-vendor interactions are not supported by default. For the purposes of this work, we focus on devices that utilize the IEEE 802.11 networking stack, as it is the most widely deployed wireless standard and underpins many vehicular and infrastructure-based communication systems [29]. We concentrate on network-level authentication for such devices, treating device integrity mechanisms (*e.g.,* attestation) as complementary.

IoT Gateways and Messaging Protocols. Despite their reliance on the Internet, IoT devices are rarely directly exposed to it. Given their constrained resources and high susceptibility to attack, these devices are typically deployed behind network gateways that provide isolation via NAT and firewalls. In smart homes, this role is usually fulfilled by a consumer-grade router, whereas in industrial and urban deployments, edge servers or aggregation points serve as IoT gateways. Device communication is facilitated by specialized messaging protocols such as MQTT, CoAP, XMPP, AMQP, and WebSockets that have been

widely adopted for IoT, offering lightweight and reliable messaging over TCP/IP and tailored to their respective deployments. These protocols are natural termination points for authenticated channels, and later sections show how they can be secured using mutual TLS between devices.

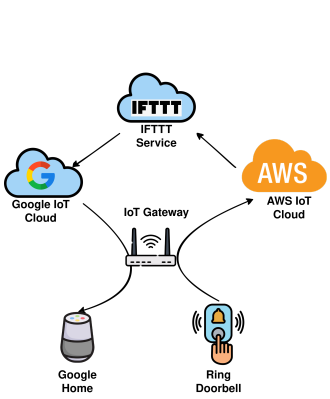
Interoperability and Latency Costs. The cloud-first model has the most visible impact in smart home environments, where vendors prioritize vertical integration and tight control over interoperability. Devices are hardcoded to interact exclusively with their respective IoT cloud services, resulting in a fragmented and siloed ecosystem. To enable cross-vendor functionality, third-party services act as intermediaries between isolated cloud platforms (Figure 1a) [30, 72]. While these services enable limited integration, they also introduce multi-hop communication paths and complex trust dependencies, challenges that have been repeatedly highlighted in prior security research [23, 14, 13, 71, 67, 34, 32]. Crucially, these integrations rely on long-lived OAuth bearer tokens [71, 14, 67, 15] that act as permanent credentials; if intercepted, they grant persistent access until manually revoked. This risk is compounded by IoT protocols (*e.g.,* MQTT, CoAP) often running over unencrypted channels within local networks [2].

Beyond these well-known security concerns, cloud-mediated architectures also impose scalability and responsiveness limits rarely addressed in prior literature. To quantify this, we measured end-to-end latency on AWS IoT under varying device loads (5–245 clients*) and message rates (5–100 msgs/sec) for a typical cross-vendor path (Device → AWS → IFTTT/Lambda → AWS → Device). As shown in Figure 1b, the distribution is heavy-tailed (Weibull, β=0.5, λ=6.37 s, mean 12.74 s), driven by throttling in the cloud integration layer—illustrating how multi-hop cloud mediation is ill-suited for latency-sensitive applications.

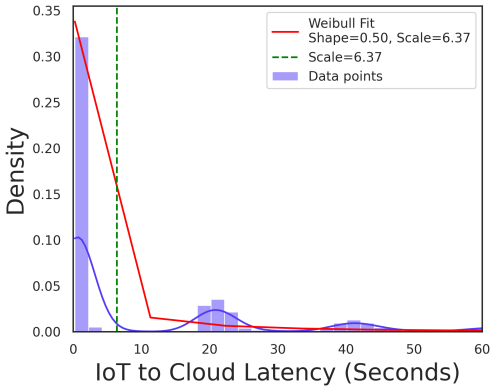
IoT PKI Today. Many IoT vendors already use PKI to manage device credentials, either through in-house infrastructure or third-party providers [16, 27, 17]. Cloud orchestration platforms such as AWS IoT Core and Azure IoT Hub [5, 18] expose these capabilities as managed services, and alliance-based efforts like Matter [31] define shared roots of trust for participating ecosystems. However, these deployments are vertically scoped: each ecosystem maintains its own trust anchors, and certificates issued within one ecosystem are not verifiable across others. As a result, there is no globally shared trust anchor for IoT devices that would enable seamless cross-vendor authentication. This fragmentation of PKI mirrors the interoperability and latency issues described above, and motivates our use of the Web PKI as a candidate shared trust infrastructure for IoT.

ACME Model. Automated Certificate Management Environment (ACME) is a protocol designed to automate the process of domain validation and certificate issuance, renewal, and revocation [11]. Developed by the Internet Engineering Task Force (IETF) and popularized by Let's Encrypt, ACME eliminates manual steps in certificate lifecycle management, thereby enhancing scalability and reducing human error. The protocol has been widely adopted by global CAs deploying it at the Internet scale [1]. Prior to ACME, obtaining and managing

*A single AWS-IoT instance seems to drop clients after 240 concurrent connections even though the developer documentations quote 500 connections per second per account is supported [4].



(a) Current cross-vendor communication approach using cloud-mediated paths.



(b) Latency distribution for cloud-mediated communication as observed in AWS IoT.

Fig. 1: Cloud-mediated IoT architectures introduce single points of failure and heavy-tailed latency (b), rendering them unsuitable for real-time cross-vendor applications.

a certificate required manual provisioning and proof-of-identity exchanges and often incurred financial costs, all of which discouraged widespread TLS adoption [42]. Today, web service providers have widely adopted the ACME model for authentication across the Internet, with Let's Encrypt alone accounting for more than 600 million active domains [26]. These properties make ACME a natural candidate for automating certificate lifecycle management for IoT devices at scale, a direction that we explore in the design of Atlas.

3 Atlas

Atlas is a PKI-based authentication framework designed to enable secure, direct communication across cross-vendor IoT devices and administrative domains. Inspired by the Web PKI model, Atlas addresses the unique challenges of IoT ecosystems by defining clear design goals, tackling technical constraints, and leveraging key insights from existing authentication frameworks.

As motivated in §1, we seek an authentication solution that integrates with prevalent IoT protocols, supports resource-constrained devices, and coexists with existing vendor cloud infrastructure. We adopt a network-centric threat model (Dolev-Yao): an in-path adversary can observe and manipulate traffic but lacks physical access to devices. Vendors are trusted within their namespace, consistent with the Web PKI model.

3.1 Design Goals

To ensure scalability and security, we identify six core principles for a cross-vendor authentication framework:

G1 Scalability Across Vendors. Support the IoT ecosystem's growth, which includes over 16 billion devices and 14,000 vendors [58], while maintaining efficiency across diverse deployment scenarios.

G2 Cross-Vendor Interoperability. Facilitate secure authentication between isolated vendor ecosystems through a unified trust model that respects vendor autonomy and data governance.

G3 Legacy Support and Integration. Seamlessly integrate with widely-used IoT protocols (*e.g.,* MQTT, CoAP) and existing infrastructure while ensuring backward compatibility with legacy devices.

G4 Security and Revocation. Provide robust security guarantees against adversarial threats, including impersonation and device compromise, with mechanisms for granular revocation.

G5 Low Overhead. Ensure computational efficiency and minimal communication latency, particularly for real-time systems, without compromising performance on resource-constrained devices.

G6 Decentralized Trust. Minimize reliance on centralized cloud infrastructure during runtime by prioritizing direct device-to-device authentication.

Table 1 compares existing approaches for IoT authentication against these design goals. See detailed analysis in §8.

3.2 Design Rationale: From Web PKI to IoT PKI

The Web PKI shows how federated trust scales across administrative domains: certificate chains anchored in globally trusted root stores decouple identity from single providers and scale to billions of endpoints without central coordination.

Naively applying this model to IoT fails: devices sit behind NAT, lack stable public endpoints, and cannot participate directly in ACME domain validation. Existing IoT PKI deployments are vertically siloed [3, 17, 31], leaving no globally trusted CA hierarchy for cross-vendor authentication.

Key Insights. To bridge this gap between Web PKI principles and IoT realities, Atlas makes three key design choices.

First, Atlas defines a vendor-rooted DNS namespace for device identities. Each vendor controls a DNS zone (*e.g.,* `*.vendor.com`), and every device is assigned a UUID-derived subdomain within that zone. This produces globally unique, resolvable identifiers that remain under vendor administrative control, and that can be embedded directly into X.509 certificates.

Second, Atlas treats the vendor IoT cloud as the ACME client and lifecycle manager rather than the device itself. The cloud infrastructure responds to ACME challenges on behalf of devices, obtains certificates from an ACME-compatible CA, and delivers them to devices over existing management channels. This design sidesteps NAT and connectivity constraints while reusing the operational patterns that vendors already employ for firmware updates and telemetry.

Third, Atlas requires that device certificates chain to a globally shared root of trust, either in the existing Web PKI or in a future IoT-specific global PKI. This ensures that any device or gateway with the corresponding root store can validate any other device's certificate and enforce mutual TLS across vendor boundaries. Because certificates share a common trust anchor, revocation information and validation logic can also be standardized and reused across the ecosystem.

These insights guide the concrete architecture presented next. §3.3 describes how we instantiate these ideas using ACME and vendor clouds, and §3.3–§3.3

Table 1: Comparison of current relevant approaches for establishing device identity and authentication in IoT.
Pairing-based Encryption (PBE) establishes symmetric keys between devices through physical or contextual proximity. Identity-based Device Authentication (IDA) uses network behavior or hardware primitives to infer device identity, but **does not** support the full authentication lifecycle in the traditional sense. Blockchain-based PKI (b-PKI) decentralizes authentication by embedding device behavior and attestation records into distributed ledgers. Traditional PKI (PKI) solutions involve widely adopted X.509 digital certificates for device authentication. See §8 for detailed analysis.
○: No Support, ◐: Unlikely Adoption / Minimal Support, ●: Readily Supports, ✔: Currently Deployed

Type	System	Scalability	Cross-Vendor Interoperability	Legacy Support	Granular Revocation	Low Runtime Overhead	Decentralized Auth
PBE	IoTCupid [22]	○	●	◐	●	●	○
	UniverSense [48]	○	◐	◐	○	◐	○
	T2Pair [39]	○	●	◐	○	◐	○
	Move2Auth [73]	○	●	◐	○	◐	○
IDA	IoT-ID [66]	○	◐	◐	●	◐	◐
	MCU-Token [69]	○	◐	◐	●	◐	◐
	Z-IoT [10]	○	○	◐	●	○	○
	IoT-Sentinel [44]	○	◐	●	●	○	○
b-PKI	Academic Proposals [45, 74, 65]	◐	●	○	◐	○	●
	Industry Proposal (Knox Matrix [40])	◐	●	○	◐	○	●
PKI	Vendor-controlled PKI [17, 16, 27] ✔	○	○	●	○	●	○
	Alliance-based PKI [5, 18, 31] ✔	◐†	◐†	●	◐†	◐†	◐†
	Atlas [Our System]	●	●	●	●	●	●

detail how identity management, CA models, revocation, and constrained devices are handled within this framework.

3.3 System Overview

Atlas extends the Web PKI model to IoT, allowing vendors to use existing Web CAs via ACME to issue globally verifiable X.509 certificates. As shown in Figure 2, Atlas integrates with IoT clouds to enable cross-vendor authentication through two key stages: *Domain Binding* and *Certificate Issuance*.

†Only supported among participating vendors. Non-participating members default to cloud-mediated setups or ad hoc solutions.

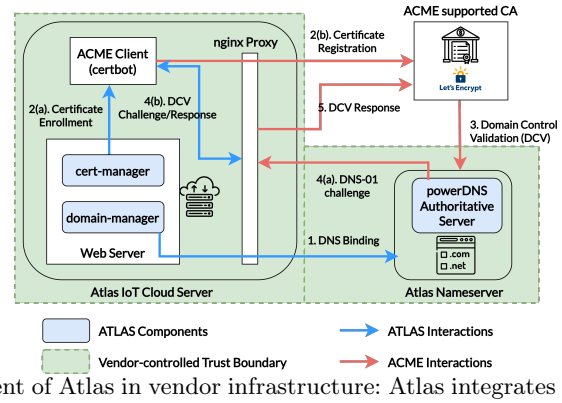


Fig. 2: Deployment of Atlas in vendor infrastructure: Atlas integrates with existing IoT Cloud setups, leveraging ACME for certificate issuance.

First, each device is assigned a globally unique UUID embedded in the vendor's DNS namespace. For example, a device with UUID *123a4567-b89c-12d3-a456-1234567890* from *vendor.com* is assigned the subdomain *<123a4567-b89c-12d3-a456-1234567890>.<vendor>.<com>*. This DNS-based identifier provides a resolvable namespace under vendor control. Second, Atlas leverages ACME for certificate issuance. Vendors operate ACME challenge responders in their IoT clouds to validate device-specific domains. The CA issues X.509 certificates binding the device's DNS record to a cryptographic identity, enabling secure device-to-device and device-to-cloud authentication.

Atlas ensures backward compatibility with existing IoT protocols and supports certificate revocation and renewal workflows. By aligning with the Web PKI model, Atlas establishes a scalable, decentralized framework for IoT authentication.

Key Management and Certificate Lifecycle. Atlas assigns each device a globally unique identifier within the vendor's DNS namespace and binds this identifier to a device-specific keypair and X.509 certificate. Vendors generate device keys during manufacturing, then use the Atlas framework to issue the corresponding certificates from an ACME-compatible CA. Certificates are short-lived and renewed periodically through the vendor's IoT cloud, which coordinates ACME interactions and delivers updated credentials to devices over existing management channels. This lifecycle design minimizes long-term exposure of keys, supports unattended operation at scale, and allows vendors to integrate Atlas with their existing provisioning and update workflows. Detailed issuance, renewal, and revocation mechanisms are described in §4.

Operational CA Models. Atlas assumes that device certificates ultimately chain to a globally shared root of trust, such as those used in the Web PKI or a future IoT-specific global PKI. Vendors may use public ACME-compatible CAs (*e.g.,* Let's Encrypt) to obtain device certificates that are immediately verifiable by any relying party with a standard root store. Alternatively, vendors or alliances may operate ACME-based intermediate CAs that are subordinate to such global roots and issue certificates within a constrained ecosystem while still following the same protocol flows. This design ensures that mutual TLS across

vendor boundaries is enforced by a common trust anchor, while still allowing flexibility in how vendors manage issuance and operational policies.

Revocation Strategy. Revocation is a critical component of any PKI-based authentication system, particularly in IoT settings where device compromise is common. Atlas relies on short-lived certificates to limit the window of exposure for stale credentials and leverages Web PKI mechanisms such as CRLs and CRLite to distribute revocation information efficiently to relying parties. Because Atlas issues certificates from globally recognized CAs, revocation events become standardized and globally actionable: any gateway or service that consumes the corresponding revocation feeds can immediately reject compromised devices, regardless of vendor. This stands in contrast to current IoT ecosystems, where bearer tokens and vendor-specific access keys must be manually decoupled from user accounts and third-party services, a process that prior work has shown to be error-prone and difficult to scale [23, 71]. We discuss this in detail in §4.4.

Support for Constrained Devices. Atlas accommodates resource-constrained devices by reusing existing TLS stacks with compact certificate chains, session resumption, and hardware acceleration where available. Our experiments confirm that even the ESP32 microcontroller sustains mTLS without performance degradation, and Atlas provides a standards-based path toward mutual authentication aligned with Zero Trust principles [60].

4 Implementation

This section details the implementation of Atlas, structured around the IoT device lifecycle. We address four core components: *Identity Creation*, *Device Binding and Enrollment*, *Renewal and Expiration*, and *Revocation*. Together, these components demonstrate how Atlas enables secure, scalable, and standards-aligned certificate management for IoT deployments.

4.1 Identity Creation

IoT devices are typically identified by MAC addresses, which are intended to be globally unique but frequently recycled by manufacturers due to acquisition costs and production constraints [9]. This makes MAC addresses unsuitable as persistent global identifiers for cross-domain authentication.

Atlas addresses this by using Universally Unique Identifiers (UUIDs) [37], which require no central authority and are widely adopted for persistent identifiers [43, 24, 8]. We use version 5 UUIDs, a name-based scheme that applies a cryptographic hash over a combination of the vendor's root domain namespace and a device-bound secret [19]. This produces a globally unique, reproducible identifier that is collision-resistant and independent from MAC reuse limitations. Table 2 formalizes our UUID derivation and namespace.

In production, the UUID derivation can use hardware-backed secrets from PUFs [41], DICE keys [63], or TEE-stored keys; Atlas requires only a stable, high-entropy, vendor-controlled identifier. Our prototype uses a synthetic MAC-derived identifier, but the framework is parametric in the identity source.

Each UUID maps to a web-resolvable subdomain under the vendor's domain, creating a structured web identity for every device. Table 2 defines a formal Uniform Resource Namespace (URN) for IoT devices using device class and

Table 2: Formal namespace for IoT devices. *Each device URN embeds a UUID derived from either our synthetic UUIDv5 generator or hardware-backed keys into a vendor-controlled DNS namespace.*

$$
\begin{aligned}
\langle \text{URN} \rangle &\models \langle \text{UUID} \rangle\ .\langle \text{device-class} \rangle\ .\langle \text{D} \rangle \\
\langle \text{UUID} \rangle &\models \mathit{UUID}_{\mathit{synth}} \mid \mathit{UUID}_{\mathit{DICE}} \mid \mathit{UUID}_{\mathit{TEE}} \\
\langle \text{UUID}_{\text{synth}} \rangle &\models \text{UUIDv5}\big(\text{NS}_{\text{DNS}}, H(D \parallel S_{\mathit{device}})\big) \\
\langle \text{UUID}_{\text{DICE}} \rangle &\models \text{UUIDv5}\big(\text{NS}_{\text{DNS}}, H(D \parallel K_{\mathit{DICE}})\big) \\
\langle \text{UUID}_{\text{TEE}} \rangle &\models \text{UUIDv5}\big(\text{NS}_{\text{DNS}}, H(D \parallel K_{\mathit{TEE}})\big) \\
\langle \text{device-class} \rangle &\models \textit{Semantic domain separator} \\
\langle \text{D} \rangle &\models \textit{Vendor-controlled DNS namespace} \\
\langle \text{NS}_{\text{DNS}} \rangle &\models \textit{UUID DNS namespace constant} \\
\langle S_{\text{device}} \rangle &\models \textit{Device-bound secret} \\
\langle K_{\text{DICE}} \rangle &\models \textit{Device key exposed by the DICE chain} \\
\langle K_{\text{TEE}} \rangle &\models \textit{Device key stored inside the TEE} \\
\langle H(\cdot) \rangle &\models \textit{Cryptographic hash function}
\end{aligned}
$$

vendor's root domain as <UUID.device-class.root-domain>, which can be used directly in X.509 certificates issued by any ACME-compatible CA.

4.2 Binding and Enrollment

Unlike individually provisioned websites, IoT devices are mass-manufactured with consistent configurations. Atlas extends the manufacturing workflow by provisioning each device with a certificate establishing a persistent digital identity (Figure 3).

The Atlas server computes a version 5 UUID for each device using its MAC address and the vendor's root domain, generating a globally unique, deterministic identifier that mitigates MAC address reuse risks. This UUID becomes the device's canonical identifier and is stored in the Atlas inventory database. The server then registers a new subdomain for the device under the vendor's domain, using the UUID as the subdomain label (*e.g., c69f00ac-532c-11ee-87f2-079e7eb2f068.camera.vendor.com*). This registration request goes to the Atlas nameserver, an authoritative nameserver handling DNS requests for the vendor's root zone. All device subdomains use `CNAME` records resolving to the vendor's IoT cloud IP, allowing the vendor's infrastructure to terminate incoming requests without exposing devices.

Atlas then initiates ACME certificate enrollment for the subdomain with any supported CA using a Certificate Signing Request (CSR). Domain control validation occurs through HTTP-01 or DNS-01 challenges executed by the vendor's ACME client. Since subdomain traffic routes to the Atlas server, it can respond to ACME challenges, proving domain control and obtaining a certificate from a trusted CA. This certificate is installed on the device during provisioning. The Atlas server maintains a metadata table with certificate enrollment timestamps, expiration dates, and renewal states for policy enforcement. Importantly, device private keys are ephemeral—generated during enrollment, injected into the device, then discarded from the server. This stateless approach to cryptographic material reduces key management overhead and improves security, and realizes the vendor-rooted identity and cloud-managed lifecycle design in §3.3.

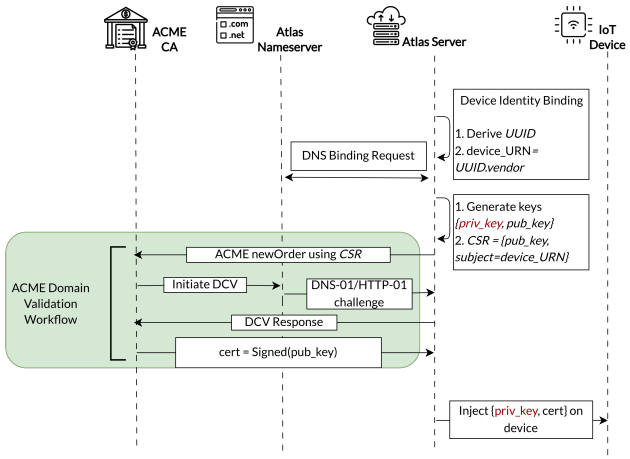


Fig. 3: *Atlas Binding and Enrollment*: Atlas performs DNS binding and ACME-based certificate enrollment during manufacturing. Device key pairs are generated for each device, injected once, and never stored persistently on the Atlas backend; any transient copies are discarded immediately after provisioning.

4.3 Renewal and Expiration

Short-lived certificates reduce the operational risk of compromised credentials, with Let's Encrypt mandating 90-day certificates to enforce cryptographic hygiene. This is especially important in IoT ecosystems where manual intervention at scale is impractical. Atlas provisions each device with a unique, domain-bound certificate from a globally trusted ACME CA that must be renewed regularly.

For devices with persistent Internet connectivity, renewals occur through an automated pipeline between the device and Atlas server via the vendor's IoT cloud. The Atlas Client–a lightweight, platform-agnostic agent–integrates with existing device software to handle certificate lifecycle management. Most IoT vendors already deploy heartbeat applications that periodically communicate with cloud infrastructure; Atlas Client extends this pattern by embedding certificate validity checks within the same control loop, triggering renewals 30 days before expiration. The client's minimal footprint allows embedding into vendor stacks with minimal engineering overhead.

For devices with intermittent connectivity, Atlas employs a proxy-based renewal mechanism. These devices typically synchronize through IoT gateways, which can mediate certificate renewals by forwarding requests to the vendor's IoT cloud. The Atlas server verifies the gateway's authorization using pre-established device-gateway bindings similar to OAuth token delegations [15]. Throughout this process, device private keys remain secure and never transmit over the network.

Unless a compromise occurs, devices retain their original 2048-bit RSA private keys from manufacturing, conforming to CA/Browser forum requirements [61]. This allows efficient certificate re-issuance while preserving cryptographic identity across renewals.

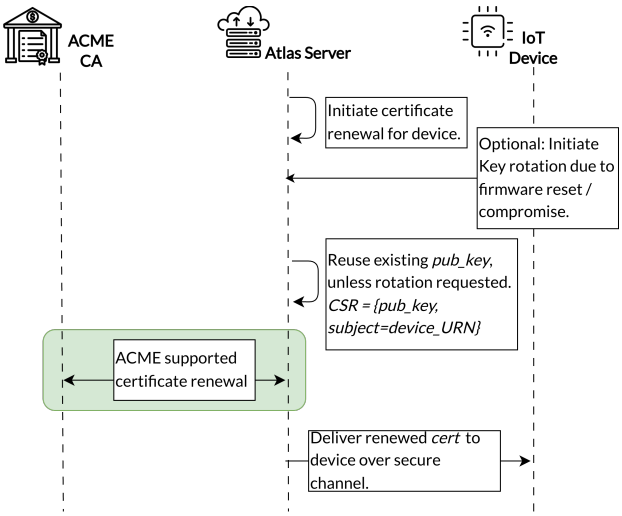


Fig. 4: Atlas Client runs alongside existing vendor software to perform certificate validity checks and automated renewal. Renewal requests reuse the device's existing public key while private key never leaves the device. When a new key pair must be issued (*e.g.,* after firmware refresh), the full enrollment protocol in Figure 3 is re-executed.

4.4 Revocation

IoT deployments are highly exposed to vulnerabilities, making certificate revocation critical for mitigating risks from device compromise [57, 7]. Traditional mechanisms like Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP) are impractical for IoT due to their size, update frequency, and network requirements [49]. Modern browser ecosystems have adopted CRLite, which compresses revocation data into probabilistic data structures (*e.g.,* Bloom filters) that can be efficiently distributed to clients [36]. Our protocol follows this approach: for each vendor, the Atlas backend periodically publishes a vendor-specific compressed CRL filter that encodes revoked device certificates. IoT gateways and backend services subscribe to these filters and perform local status checks during the TLS handshake, terminating connections from compromised devices without contacting the CA online. This realizes the globally actionable revocation strategy described in §3.3 while ensuring that device private keys never leave the device.

Our implementation totals approximately 2.5K lines of code across Python, C++, Arduino, and Shell scripting, and is open-sourced for future research [6].

5 Evaluation

We now evaluate Atlas to assess its viability, performance, and comparative advantage. Three research questions guide our evaluation: **RQ1** (Device Feasibility): Is Atlas practical on resource-constrained IoT hardware? **RQ2** (Provisioning Scalability): What overhead does Atlas introduce on vendor cloud infrastructure at fleet scale? **RQ3** (Application Impact): Does Atlas improve end-to-end latency and scalability compared to cloud-mediated deployments? Security analysis is deferred to §6.

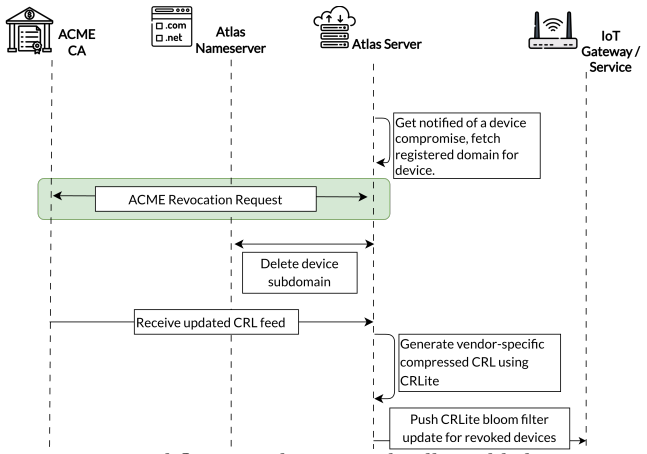


Fig. 5: Atlas revocation workflow: vendors periodically publish compressed revocation filters that encode revoked device certificates. Any IoT gateway or service that terminates device connections can subscribe to these filters and locally reject handshakes from compromised devices, without ever accessing device private keys.

5.1 Experimental Setup

We implement Atlas across three layers of the IoT stack. *Devices*: our testbed includes a Raspberry Pi 4 (RPi4) [54], Raspberry Pi Zero W (RPi0W) [55], and ESP32 [20] development board, covering a spectrum from multi-core Linux-class systems to microcontrollers, all equipped with IEEE 802.11 wireless networking stack. *IoT Cloud*: we deploy a Atlas-compatible vendor cloud instance on a Ubuntu 20.04 server that integrates HTTPS endpoints, an application backend, an authoritative DNS server for managing per-device subdomains, and an ACME client for certificate lifecycle management (*e.g.,* certbot); for comparison, we also use AWS IoT Core as a representative cloud-mediated baseline. *IoT Ecosystems*: we evaluate Atlas in a local smart home testbed with an MQTT broker [25] and in a smart city scenario modeled in ns-3 [46], where mobile IoT nodes communicate with static gateways, and cloud-mediated latencies are emulated using an empirically derived Weibull distribution.

5.2 Device Performance

We quantify the impact of Atlas on device-level performance and Quality of Service (QoS) by examining latency and CPU utilization across representative IoT hardware platforms under varying operational loads (**RQ1**). These metrics allow us to assess whether cross-vendor authentication via Atlas (using mTLS) introduces overheads that could impair typical IoT device operations.

Our experimental design reflects the diversity of IoT messaging use cases. We focus on short-length, high-frequency communication patterns that are common in telemetry, actuation, and sensor reporting scenarios. To ensure broad applicability, we perform our experiments on three embedded systems: RPi4, RPi0W, and ESP32, reflecting the architectural capabilities of standard IoT devices. We implement a local testbed consisting of a publisher-subscriber model, where the subscriber is fixed, and the publisher is varied across the three devices. An RPi4

Table 3: Latency measurements show that the overhead of TLS is well within the real-time performance thresholds for IoT. Modern micro-controllers (*e.g.,* esp32) have dedicated cryptographic accelerators that offload TLS computation.

Metric		**rpi4**	**rpi0**	**esp32**
Latency (ms)	tcp	27.68	29.25	56.54
	tls	44.28	39.47	80.76
CPU (%)	tcp	4.80	7.28	18.69
	tls	8.94	7.69	10.56

hosts the local MQTT broker (using Eclipse Mosquitto[25]), emulating a typical automation hub setup. MQTT is chosen due to its lightweight nature and popularity as the de facto IoT messaging protocol.

Each publisher sends messages of 256 bytes at configurable rates (5 to 100 messages per second), holding each rate for 5 minutes. We record end-to-end latency and CPU utilization on the publisher under two configurations: (1) MQTT over TCP (insecure) and (2) MQTT over mTLS (cross-vendor authentication via Atlas). This setup allows us to isolate the cost of secure channel establishment and sustained cryptographic processing required for applications using the Atlas framework. For the ESP32, we implemented MQTT over TLS instead of full mutual TLS due to a known limitation in the underlying mbedTLS library [51]. However, we note that mTLS is simply TLS with additional client certificate verification done by the server, establishing a functionally equivalent secure channel for the purposes of this evaluation.

Findings: Figure 6 compares latency and CPU utilization across varying message rates for each platform. Table 3 presents a consolidated summary, reporting mean end-to-end latency and average CPU load for each device under both insecure (MQTT over TCP) and secure (MQTT over mTLS) configurations.

Takeaways: Secure channel establishment adds only ∼17 ms mean latency across devices (16 ms on RPi4, 10 ms on RPi0W, 24 ms on ESP32)—well within real-time IoT thresholds. CPU overhead is marginal ($\leq$4.14% increase on RPi4 and RPi0W), benefiting from lightweight TLS library optimizations. Notably, on the ESP32, TLS *reduces* CPU utilization by 8 percentage points relative to plain TCP (10.56% vs. 18.69%, Table 3). This counter-intuitive result stems from the ESP32's dedicated hardware cryptographic accelerator (AES-128/256, SHA), which mbedTLS fully exploits via Espressif's hardware-backed port [68]. The accelerator processes TLS record encryption and decryption via DMA-based transfers, so the Xtensa LX6 CPU is not stalled during crypto operations. In pure TCP mode, the CPU handles all data processing itself; in TLS mode, the hardware absorbs the per-packet crypto load, making encryption effectively transparent to the CPU and yielding lower CPU utilization than plain TCP.

5.3 IoT Cloud Overhead

For **RQ2**, we focus on quantifying the latency introduced by Atlas during device provisioning at scale, particularly evaluating its implications for certificate lifecycle management. IoT vendors require enrollment processes that are fast, scalable, and automated, as manual or slow provisioning directly hinders manufacturing throughput and product deployment timelines. Using our experimen-

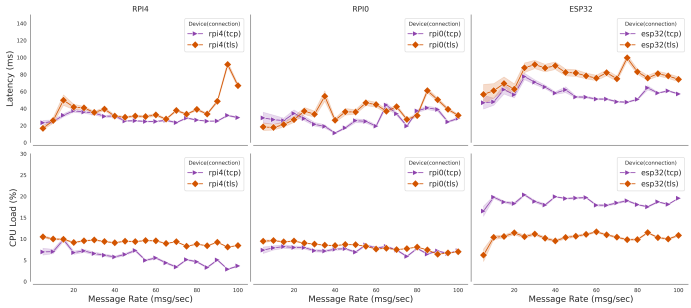


Fig. 6: Impact of mTLS on Resource-Constrained Devices: Atlas introduces negligible latency (∼17ms) and CPU overhead (<9%), demonstrating feasibility even on microcontrollers (ESP32).

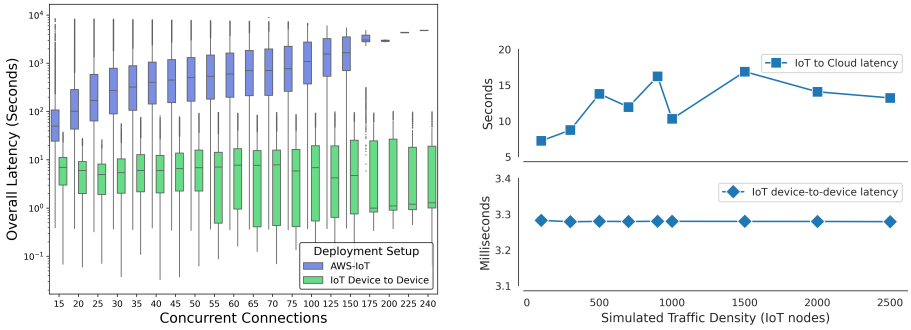
tal IoT cloud setup (Figure 2), we measure the latency of the two critical operations in Atlas provisioning: (1) Domain Binding, where a unique device identity is associated with a DNS namespace under the vendor's zone, and (2) Certificate Issuance, where Atlas invokes the ACME protocol to validate domain ownership and retrieve a device certificate. We use Let's Encrypt as the ACME-compatible CA, which operates ACME at the Internet scale and supports short-lived certificate issuance. To emulate bulk provisioning scenarios, we conducted enrollment batches ranging from 50 to 500 devices. *Domain Binding* operations were performed using our authoritative DNS server, while *Certificate Issuance* was executed against the Let's Encrypt ACME staging endpoint [38]. The staging CA mirrors Let's Encrypt's production setup and enforces equivalent rate limits—allowing up to $30,000$ certificate requests per week per registered domain.

Findings and Takeaway: The Atlas framework incurs an average of 0.03 ± 0.008 seconds per device for Domain Binding and 4.85 ± 1.36 seconds per device for Certificate Issuance. These costs are incurred primarily at manufacturing time or during scheduled renewals and demonstrate that ACME-based provisioning is fast enough to support fleet-scale deployments without becoming a bottleneck for vendors.

5.4 IoT Applications Using Atlas Framework

In order to quantify the performance benefits of using Atlas in real IoT applications (**RQ3**), we design and execute two sets of experiments using the smart home and smart city setups described in §5.1.

Experiment 1: Smart Home Automation. We begin with our smart home testbed to evaluate the end-to-end latency in typical IoT messaging scenarios. A local Mosquitto MQTT broker is used to implement a trigger-action automation system within a LAN environment. For comparison, we replicate the same functionality using AWS IoT infrastructure, where Raspberry Pi 4 (RPi4) devices, provisioned with the AWS IoT SDK, act as publishers and subscribers. The cloud-hosted trigger-action logic is implemented using AWS Lambda. This baseline mirrors the current IoT cloud architecture, as previously illustrated in Figure 1a. Our measurements directly compare the two deployment paradigms: (1)



(a) Smart home workload (RPi4). (b) Smart city workload (ns-3 simulation).

Fig. 7: Performance comparison under load. (a) Smart Home: Atlas maintains stable, low latency while AWS IoT exhibits high variance and packet drops. (b) Smart City: Atlas scales to thousands of nodes with millisecond-level latency, contrasting with the heavy-tailed, multi-second delays of cloud mediation.

secure, direct D2D applications using Atlas, and (2) traditional cloud-mediated applications through AWS IoT.

Experiment 2: Smart City Automation. To evaluate scalability, we use our ns-3 smart city simulation (full setup in Appendix C), modeling mobile IoT nodes (50–2500, *e.g.,* autonomous vehicles) that periodically communicate with five static IoT gateways (*e.g.,* traffic management servers) via D2D or simulated cloud-mediated paths.

To model cloud-mediated latency, we inject a synthetic Weibull delay with parameters β=0.5, λ=6.37 s empirically derived from our AWS IoT measurements (Figure 1b).

Findings: Figure 7a demonstrates the performance of smart home IoT applications using Atlas framework under a scaled workload, while Figure 7b illustrates the latency distribution observed in large-scale smart city-style deployments powered by Atlas versus a simulated cloud-mediated baseline.

Takeaway: Our results indicate that traditional IoT cloud infrastructures are only feasible in limited contexts: primarily single-tenant, latency-tolerant smart home environments with low message throughput and tightly coupled vendor ecosystems. However, as communication demands scale—either through higher message rates or increased device count—cloud-mediated architectures exhibit significant performance degradation. In *Experiment 1*, the AWS IoT setup displayed substantial latency variance under moderate load and dropped most packets beyond 150 concurrent connections, even though developer documentations claim that AWS-IoT can support higher throughput (upto 500 concurrent connections [4]). In *Experiment 2*, cloud-mediated communication regularly exceeded 10 seconds of delay per device, even for basic message exchanges. In contrast, IoT applications using Atlas demonstrate consistent, low-latency communication making it suitable for real-time and safety-critical systems.

Limitations. Our measurements cover steady-state performance; fault-injection studies, CA rate-limit stress tests, and full mTLS on ESP32 are left to future work (see Appendix 7 for details).

6 Qualitative Security Analysis of Atlas

6.1 Atlas End-to-End Security

We analyze the end-to-end security of the Atlas framework under the Dolev-Yao threat model (Section 1), wherein the adversary can intercept, modify, inject, replay, or drop any network message. Our analysis considers all critical components of Atlas: IoT Cloud, ACME CA, and IoT Device, and evaluates how the design is robust against the canonical classes of Dolev-Yao attacks.

Trusted Infrastructure. The IoT Cloud includes the vendor-operated web server and authoritative nameserver, both within the vendor's trust boundary and protected via standard practices (firewall enforcement, TLS-only communication, intrusion prevention via fail2ban [21]). Internal communications use static IP routing, eliminating DNS-based resolution within vendor infrastructure and preventing MitM and DNS poisoning attacks. Atlas deploys PowerDNS [53] as its authoritative nameserver, enabling real-time subdomain issuance and granular TTL configuration without registrar rate limits or propagation delays. The DNS zone is tightly controlled and accessible only by authenticated administrators.

CA Interactions. Certificate issuance in Atlas is handled using the ACME protocol, which has been formally verified for security [12]. The client (*e.g.,* Let's Encrypt's certbot) interacts with the CA over HTTPS for control messages including certificate requests, challenge submission, and certificate downloads. These HTTPS interactions ensure protection against MitM, impersonation, and downgrade attacks. Domain validation itself (*e.g.,* via http-01) is performed over unauthenticated HTTP so that the CA can validate actual control over the domain by probing a specific token served by the device or cloud server. While HTTP is used for this validation step, the challenge itself is cryptographically bound to the client's ACME account key, and includes fresh, signed tokens. This mechanism prevents spoofing, replay, and reflection attacks. Only clients who can both respond to the challenge and prove possession of the private key can successfully obtain a certificate.

Device Certificate Provisioning. Each IoT device is provisioned during manufacturing with a unique X.509 certificate and keypair. This step occurs under vendor control and ensures that each device has a globally unique and verifiable cryptographic identity. Certificate renewals are handled periodically via over-the-air (OTA) updates, which are conducted over TLS channels and authenticated by both client and server using mutual TLS (mTLS). This setup ensures that all subsequent device communications are cryptographically authenticated and encrypted, providing protection against eavesdropping, impersonation, injection, and message tampering. TLS session freshness and challenge-response authentication also prevent replay attacks. The use of TLS 1.3 and controlled cipher suite negotiation ensures that protocol downgrade attacks are not feasible.

6.2 Additional Privacy and Security Considerations

Privacy Issues. Device subdomains appearing in Certificate Transparency (CT) logs may expose metadata about a vendor's deployment or device inventory, raising competitive privacy concerns. However, Atlas provides multiple mitigations. First, CT entries do not indicate whether a certificate is ever used or deployed, allowing vendors to generate excess certificates as decoys. Second, device

URNs are constructed to allow anonymized identifiers, such as UUIDs, without semantic information about device types or capabilities. Finally, vendors with stricter privacy requirements can operate their own ACME-compatible intermediate CA, bypassing public CT altogether while remaining compliant with Atlas's operational model.

Malicious Vendor. Atlas assumes that vendors will faithfully participate in the certificate lifecycle, including proper key generation, renewal, and revocation. A malicious or negligent vendor could, in theory, retain copies of device private keys or ignore renewal requirements, undermining security guarantees. However, this threat is not unique to Atlas; all modern consumer devices rely on vendors for essential security updates. Moreover, a vendor already has complete control over its devices in vertically integrated systems, so Atlas does not introduce new incentives or opportunities for abuse. Rather, it provides a structured, auditable mechanism for credential management at scale.

Domain Impersonation. A legitimate concern is whether adversaries can abuse Atlas's domain-based identity model via techniques like typosquatting [62], homograph attacks [56], or acquiring expired vendor domains [59]. Atlas explicitly mitigates these risks by issuing certificates only for subdomains that are cryptographically bound to an active, ACME-validated parent domain controlled by the vendor. Device identities are embedded in structured URN namespaces derived from UUIDs, making them unforgeable without domain control. Even if CT logs expose device subdomains, those entries always resolve to vendor-managed infrastructure.

Compromised Private Key. IoT devices are known to be vulnerable due to their constrained resources and limited physical protections [57, 7]. Unlike deployments with certificate reuse or hardcoded credentials [33], Atlas confines the scope of any single compromise: each device has a unique key and certificate, so one compromise does not affect the broader ecosystem. Atlas facilitates post-compromise mitigation via certificate revocation (§4.4); detection and prevention are orthogonal, addressable by complementary attestation or behavioral monitoring [44, 47].

7 Discussion

Backwards Compatibility. Vendors can integrate Atlas without overhauling their backends: certificate installation occurs on-device via OTA firmware updates, enabling a graceful migration while preserving compatibility with legacy device management tools.

Incentive Compatibility. Many ACME-compatible CAs offer zero-cost certificate issuance [1], so Atlas inherits zero-cost, highly automated, formally-verified certificate management—a compelling alternative to the fragmented, proprietary provisioning systems prevalent in IoT today.

IoT-specific PKI or not. Our evaluation suggests existing ACME deployments can already support IoT dual use. Atlas is intentionally CA-agnostic: vendors may use public Web CAs, IoT-specific intermediate CAs beneath a shared root, or private ACME-compatible CAs within alliances. As ACME adoption grows, additional hierarchies will emerge to serve IoT workloads while preserving the globally verifiable roots that Atlas relies on [1].

Limitations and Threats to Validity. Our evaluation is subject to several limitations. First, we do not empirically study availability under CA or cloud outages; our measurements capture performance in steady-state operation, and we leave fault-injection studies to future work. Second, while we analytically discuss CT privacy and CA rate limits in the design and security analysis, we do not stress-test these dimensions experimentally. Third, our ESP32 experiments use TLS rather than full mTLS due to a known library limitation [51]; however, since the client still performs certificate validation and the underlying cryptographic workloads are comparable, we expect the latency and CPU overhead conclusions to generalize to mTLS-capable firmware.

8 Related Work

Existing IoT authentication frameworks span four categories—device pairing, identity derivation, blockchain-based PKI, and traditional PKI—none of which simultaneously achieves the scalability, cross-vendor interoperability, lifecycle management, and decentralized trust in Table 1.

Pairing-based Encryption (PBE). Pairing-based schemes [73, 39, 22] establish symmetric keys via human or contextual interaction. They suit small personal deployments but do not scale: each association requires physical proximity or user effort. Atlas moves identity establishment into the network fabric (DNS and PKI), enabling unattended factory-scale provisioning (G1, G2, G3).

Identity-based Device Authentication (IDA). Fingerprinting approaches (IoT-Sentinel, IoTFinder, Z-IoT, PUF-based) [44, 50, 10, 41, 69] infer identifiers from traffic or hardware. Effective for anomaly detection, they lack stable revocable identities. Atlas complements such techniques by binding devices to globally verifiable X.509 identities with a full certificate lifecycle (G3, G4).

Blockchain-based PKI (b-PKI). Proposals such as SerenIoT, LegIoT, and Knox Matrix [64, 45, 40] offer decentralized auditability via distributed ledgers, but introduce consensus overhead and have seen limited real-world deployment on constrained hardware. Atlas reuses existing DNS and ACME federated trust, achieving interoperability and global revocation (G1, G2, G4, G6) without new consensus infrastructure.

Traditional and Cloud-Hosted PKI (PKI). Commercial PKI and IoT cloud platforms [16, 27, 17, 5, 18, 31] provide strong per-vendor authentication but remain siloed: cross-vendor interactions fall back to cloud mediation. Atlas roots device identities in vendor-controlled DNS namespaces under a globally shared CA hierarchy, enabling any compliant gateway to validate certificates across vendors while vendors retain lifecycle control (G2, G4, G6).

9 Conclusion

In this paper, we presented Atlas, a unified authentication framework that enables secure, cross-vendor device-to-device communication by extending ACME and globally shared CAs to vendor-controlled DNS namespaces for IoT devices. Atlas supports the full certificate lifecycle, including issuance, renewal, and revocation, without requiring fundamental architectural changes from vendors, while enabling mutual TLS between heterogeneous devices under a network-centric threat model. Our implementation and evaluation on constrained hardware, a

Atlas-compatible cloud stack, and smart home and smart city workloads show that Atlas is feasible and scalable in practice, delivering mTLS-based authentication with modest overhead and significantly lower, more predictable latency than cloud-mediated baselines; by enabling direct D2D communication, Atlas can improve service availability during Internet or cloud disruptions by avoiding dependence on continuous cloud mediation.

References

1. acmeclients.com: Acme certificate authoritiesfacility authorities (2025), https://acmeclients.com/certificate-authorities/, last accessed: 2025-05-26
2. Alrawi, O., Lever, C., Antonakakis, M., Monrose, F.: Sok: Security evaluation of home-based iot deployments. In: 2019 IEEE symposium on security and privacy (sp) (2019)
3. Amazon: Amazon private certificate authority pricing (2024), https://aws.amazon.com/private-ca/pricing/
4. Amazon Web Services: Aws iot core throttling limits. https://docs.aws.amazon.com/general/latest/gr/iot-core.html (2022)
5. Amazon Web Services, I.: Aws iot core documentation (2023), https://docs.aws.amazon.com/iot/latest/developerguide/manual-cert-registration.html, accessed: Oct 6, 2023
6. Anonymous: Towards secure and scalable cross-vendor authentication in performance-critical iot systems. https://anonymous.4open.science/r/atlas-1F6D/ (2025), accessed: April 15, 2025
7. Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., et al.: Understanding the mirai botnet. In: 26th USENIX security symposium (USENIX Security 17) (2017)
8. Archive, T.: Snowflake: A network service for generating unique id numbers at high scale (2024), https://github.com/twitter-archive/snowflake, repository archived on Sep 18, 2021
9. Association, I.S.: Ieee registration authority faqs. https://standards.ieee.org/faqs/regauth/ (2023), accessed: 2023-09-14
10. Babun, L., Aksu, H., Ryan, L., Akkaya, K., Bentley, E.S., Uluagac, A.S.: Z-iot: Passive device-class fingerprinting of zigbee and z-wave iot devices. In: ICC 2020-2020 IEEE International Conference on Communications (ICC) (2020)
11. Barnes, R., Hoffman-Andrews, J., McCarney, D., Kasten, J.: Automatic Certificate Management Environment (ACME). RFC 8555, Internet Engineering Task Force (March 2019), available at: https://www.rfc-editor.org/rfc/rfc8555
12. Bhargavan, K., Bichhawat, A., Do, Q.H., Hosseyni, P., Küsters, R., Schmitz, G., Würtele, T.: An in-depth symbolic security analysis of the acme standard. In: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. pp. 2601–2617 (2021)
13. Chen, Y., Alhanahnah, M., Sabelfeld, A., Chatterjee, R., Fernandes, E.: Practical data access minimization in {Trigger-Action} platforms. In: 31st USENIX Security Symposium (USENIX Security 22) (2022)
14. Chen, Y., Chowdhury, A.R., Wang, R., Sabelfeld, A., Chatterjee, R., Fernandes, E.: Data privacy in trigger-action systems. In: 2021 IEEE Symposium on Security and Privacy (SP) (2021)
15. Community, O.: Oauth 2.0. https://oauth.net/2/ (2023), accessed: Oct 12, 2023
16. Corporation, E.: Digital security solutions. https://www.entrust.com/ (2023)

17. Corporation, I.T.: ipki for iot, the private managed certificate authority for iot. https://www.intertrust.com/pki-for-iot/ (2023)
18. Corporation, M.: Manage iot edge certificates - azure iot edge (2023), https://learn.microsoft.com/en-us/azure/iot-edge/how-to-manage-device-certificates?view=iotedge-1.4&tabs=ubuntu, accessed: Oct 6, 2023
19. Davis, K., Peabody, B., Leach, P.: Rfc 9562: Universally unique identifiers (uuids) (2024), https://www.rfc-editor.org/rfc/rfc9562.html, category: Standards Track
20. Espressif Systems: ESP32-WROOM-32 Datasheet. https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (2023), accessed: 2025-04-14
21. Fail2Ban Community: Fail2ban: Daemon to ban hosts that cause multiple authentication errors (2025), https://github.com/fail2ban/fail2ban, accessed: 2025-05-29
22. Farrukh, H., Ozmen, M.O., Ors, F.K., Celik, Z.B.: One key to rule them all: Secure group pairing for heterogeneous iot devices. In: IEEE Symposium on Security and Privacy (SP) (2023)
23. Fernandes, E., Rahmati, A., Jung, J., Prakash, A.: Decentralized action integrity for trigger-action iot platforms. In: Proceedings 2018 Network and Distributed System Security Symposium (2018)
24. Forum, U.: Unified extensible firmware interface (uefi) specification. https://uefi.org/specifications (2023), accessed: Oct 12, 2023
25. Foundation, E.: Eclipse mosquitto. https://mosquitto.org/ (2023), accessed: Oct 6, 2023
26. Group, I.S.R.: Let's encrypt stats. https://letsencrypt.org/stats/ (2023)
27. Group, T.: Iot security solutions: Bringing trust to the internet of things. https://cpl.thalesgroup.com/industry/iot-security (2023)
28. Hristozov, S., Heyszl, J., Wagner, S., Sigl, G.: Practical runtime attestation for tiny iot devices. In: NDSS Workshop on Decentralized IoT Security and Standards (DISS). vol. 18 (2018)
29. IEEE: IEEE Standard for Information Technology—Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks—Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Tech. Rep. IEEE Std 802.11-2016, IEEE (2016)
30. IFTTT, I.: Ifttt. https://ifttt.com (2022), [Online; accessed 20-April-2024]
31. IoT, C.: Matter - csa iot. https://csa-iot.org/all-solutions/matter/ (2023), accessed: 2023-10-12
32. Jakaria, M., Huang, D.Y., Das, A.: Connecting the dots: Tracing data endpoints in iot devices. Proceedings on Privacy Enhancing Technologies (PoPETs) **2024**(3) (2024)
33. Kilgallin, J., Vasko, R.: Factoring rsa keys in the iot era. In: 2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA) (2019)
34. Kumar, D., Shen, K., Case, B., Garg, D., Alperovich, G., Kuznetsov, D., Gupta, R., Durumeric, Z.: All things considered: An analysis of {IoT} devices on home networks. In: 28th USENIX security symposium (USENIX Security 19) (2019)
35. Kumar, S., Hu, Y., Andersen, M.P., Popa, R.A., Culler, D.E.: {JEDI}:{Many-to-Many}{End-to-End} encryption and key delegation for {IoT}. In: 28th USENIX security symposium (USENIX Security 19). pp. 1519–1536 (2019)

36. Larisch, J., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Wilson, C.: Crlite: A scalable system for pushing all tls revocations to all browsers. In: 2017 IEEE Symposium on Security and Privacy (SP) (2017)
37. Leach, P., Mealling, M., Salz, R.: A universally unique identifier (uuid) urn namespace. Tech. rep. (2005)
38. Let's Encrypt: Staging Environment Documentation. `https://letsencrypt.org/docs/staging-environment/` (2025), accessed: 2025-04-14
39. Li, X., Zeng, Q., Luo, L., Luo, T.: T2pair: Secure and usable pairing for heterogeneous iot devices. In: Proceedings of the 2020 acm sigsac conference on computer and communications security (2020)
40. Lumb, D.: Samsung's knox matrix explained: Smart home security via private blockchain. `https://www.cnet.com/tech/mobile/samsungs-knox-matrix-explained-smart-home-security-via-private-blockchain/` (2022), accessed: September 14, 2023
41. Mall, P., Amin, R., Das, A.K., Leung, M.T., Choo, K.K.R.: Puf-based authentication and key agreement protocols for iot, wsns, and smart grids: A comprehensive survey. IEEE Internet of Things Journal **9**(11) (2022)
42. Manousis, A., Ragsdale, R., Draffin, B., Agrawal, A., Sekar, V.: Shedding light on the adoption of let's encrypt. arXiv preprint arXiv:1611.00469 (2016)
43. Microsoft: Interface pointers and interfaces (2023), `https://learn.microsoft.com/en-us/windows/win32/com/interface-pointers-and-interfaces?redirectedfrom=MSDN`, accessed: September 14, 2023
44. Miettinen, M., Marchal, S., Hafeez, I., Asokan, N., Sadeghi, A.R., Tarkoma, S.: Iot sentinel: Automated device-type identification for security enforcement in iot. In: 2017 IEEE 37th international conference on distributed computing systems (ICDCS) (2017)
45. Neureither, J., Dmitrienko, A., Koisser, D., Brasser, F., Sadeghi, A.R.: Legiot: Ledgered trust management platform for iot. In: Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part I 25 (2020)
46. ns-3 Consortium: ns-3: Network Simulator. `https://www.nsnam.org/` (2025), accessed: 2025-04-13
47. OConnor, T., Mohamed, R., Miettinen, M., Enck, W., Reaves, B., Sadeghi, A.R.: Homesnitch: behavior transparency and control for smart home iot devices. In: Proceedings of the 12th conference on security and privacy in wireless and mobile networks (2019)
48. Pan, S., Ruiz, C., Han, J., Bannis, A., Tague, P., Noh, H.Y., Zhang, P.: Universense: Iot device pairing through heterogeneous sensing signals. In: Proceedings of the 19th International Workshop on Mobile Computing Systems & Applications (2018)
49. Paracha, M.T., Dubois, D.J., Vallina-Rodriguez, N., Choffnes, D.: Iotls: Understanding tls usage in consumer iot devices. In: Proceedings of the 21st ACM Internet Measurement Conference (2021)
50. Perdisci, R., Papastergiou, T., Alrawi, O., Antonakakis, M.: Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis. In: 2020 IEEE European Symposium on Security and Privacy (EuroS&P) (2020)
51. Peskine, G.: TLS 1.2 client with client certificate sometimes uses the wrong hash algorithm in Finished. `https://github.com/Mbed-TLS/mbedtls/issues/6566` (2022), gitHub Issue #6566, Mbed TLS repository, Accessed: 2025-04-14
52. Petzi, L., Yahya, A.E.B., Dmitrienko, A., Tsudik, G., Prantl, T., Kounev, S.: Scraps: Scalable collective remote attestation for pub-sub iot networks with un-

trusted proxy verifier. In: 31st USENIX Security Symposium (USENIX Security 22) (2022)
53. PowerDNS.COM BV and contributors: PowerDNS: Authoritative Server, Recursor, and dnsdist. https://github.com/PowerDNS/pdns (2025), accessed: 2025-05-30
54. Raspberry Pi Ltd: Raspberry Pi 4 Model B Specifications. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (2022), accessed: 2025-04-14
55. Raspberry Pi Ltd: Raspberry Pi Zero W Specifications. https://www.raspberrypi.com/products/raspberry-pi-zero-w/ (2023), accessed: 2025-04-14
56. Roberts, R., Goldschlag, Y., Walter, R., Chung, T., Mislove, A., Levin, D.: You are who you appear to be: A longitudinal study of domain impersonation in tls certificates. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)
57. Ronen, E., Shamir, A., Weingarten, A.O., O'Flynn, C.: Iot goes nuclear: Creating a zigbee chain reaction. In: 2017 IEEE Symposium on Security and Privacy (SP) (2017)
58. Sinha, S.: State of iot 2024: Number of connected iot devices growing 13% to 18.8 billion globally. Available at: https://iot-analytics.com/number-connected-iot-devices/ (2024), accessed: March 2025
59. So, J., Miramirkhani, N., Ferdman, M., Nikiforakis, N.: Domains do change their spots: Quantifying potential abuse of residual trust. In: 2022 IEEE Symposium on Security and Privacy (SP) (2022)
60. Stafford, V.: Zero trust architecture. NIST special publication **800** (2020)
61. of Standards, N.I., Technology: Recommendation for key management: Part 1 - general. Tech. Rep. NIST Special Publication 800-57 Part 1 Revision 5, National Institute of Standards and Technology (2020), https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf
62. Szurdi, J., Kocso, B., Cseh, G., Spring, J., Felegyhazi, M., Kanich, C.: The long {"Taile"} of typosquatting domain names. In: 23rd USENIX Security Symposium (USENIX Security 14) (2014)
63. Tao, Z., Rastogi, A., Gupta, N., Vaswani, K., Thakur, A.V.: {DICE*}: A formally verified implementation of {DICE} measured boot. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 1091–1107 (2021)
64. Thomasset, C., Barrera, D.: Sereniot: distributed network security policy management and enforcement for smart homes. In: Proceedings of the 36th Annual Computer Security Applications Conference (2020)
65. Toorani, M., Gehrmann, C.: A decentralized dynamic pki based on blockchain. In: Proceedings of the 36th annual ACM symposium on applied computing. pp. 1646–1655 (2021)
66. Vaidya, G., Nambi, A., Prabhakar, T., Sudhakara, S., et al.: Iot-id: A novel device-specific identifier based on unique hardware fingerprints. In: 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI) (2020)
67. Wang, Q., Datta, P., Yang, W., Liu, S., Bates, A., Gunter, C.A.: Charting the attack surface of trigger-action iot platforms. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (2019)
68. wolfSSL: wolfcrypt espressif port - readme. Available at: https://github.com/wolfSSL/wolfssl/blob/master/wolfcrypt/src/port/Espressif/README.md (2025), accessed: 11 April 2025

69. Xiao, Y., He, Y., Zhang, X., Wang, Q., Xie, R., Sun, K., Xu, K., Li, Q.: From hardware fingerprint to access token: Enhancing the authentication on iot devices. In: Network and Distributed Security Symposium (NDSS 2024) (2024)
70. Xu, M., Huber, M., Sun, Z., England, P., Peinado, M., Lee, S., Marochko, A., Mattoon, D., Spiger, R., Thom, S.: Dominance as a new trusted computing primitive for the internet of things. In: 2019 IEEE Symposium on Security and Privacy (SP) (2019)
71. Yuan, B., Jia, Y., Xing, L., Zhao, D., Wang, X., Zhang, Y.: Shattered chain of trust: Understanding security risks in cross-cloud iot access delegation. In: 29th USENIX security symposium (USENIX security 20) (2020)
72. Zapier: Zapier website (2025), {https://zapier.com/}, accessed: 11 April 2025
73. Zhang, J., Wang, Z., Yang, Z., Zhang, Q.: Proximity based iot device authentication. In: IEEE INFOCOM 2017-IEEE conference on computer communications (2017)
74. Zhu, X., Badr, Y.: Identity management systems for the internet of things: a survey towards blockchain solutions. Sensors **18**(12) (2018)

A Generative AI Usage Considerations

We used GenAI tools, such as ChatGPT, Claude, and Gemini, to polish the text in this paper. The authors wrote the entire original draft first and lightly used GenAI tools to polish the text, primarily focusing on the Introduction section. Any modified text was carefully reviewed for accuracy and correctness.

B Atlas Certificate Lifecycle

B.1 Validation

Atlas provisions each IoT device with a globally valid, device-specific certificate from a publicly trusted CA, enabling direct mutual authentication via mTLS without cloud mediation. As shown in Figure 8, two devices—potentially from different vendors—can securely establish end-to-end channels within a local network.

Unlike traditional cloud-mediated setups that introduce latency and centralize trust, Atlas leverages the shared trust anchor of the Web PKI to support cross-vendor interoperability. Since all device certificates descend from a public CA, devices can independently validate each other's credentials. The mTLS handshake uses standard TLS libraries (*e.g.,* OpenSSL, mbedTLS), which verify the peer's certificate chain, validity, and revocation status against system-trusted roots. Certificates issued by Atlas embed vendor-controlled subdomains in the CN and SAN fields, offering a verifiable namespace for device identity.

By aligning with global PKI semantics, Atlas enables scalable, standards-compliant device-to-device authentication, eliminating the need for vendor-specific authorization mechanisms or out-of-band pairing.

C Experimental Setup

We implement Atlas across three distinct components—IoT Device, IoT Cloud, and IoT Ecosystem—to comprehensively assess its impact across the entire IoT stack.

IoT Devices. We use the Raspberry Pi 4 (RPi4)[54], Raspberry Pi Zero W (RPi0W)[55], and ESP-WROOM-32 (ESP32)[20] development boards for our

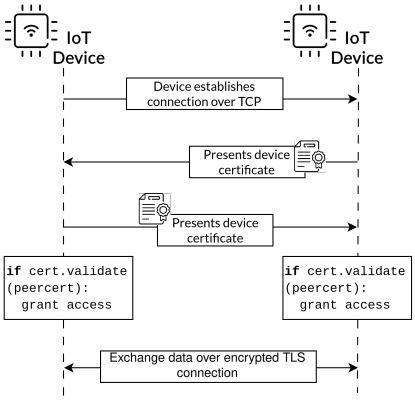


Fig. 8: Cross-Vendor Authentication using Atlas: Atlas provides a model for IoT applications to adopt cross-vendor D2D authentication using mTLS.

Table 4: Device Characteristics

Characteristic	**RPi4**	**RPi0W**	**ESP32**
Clock Speed	1.4 GHz	1.0 GHz	240 MHz
RAM	1 GB	512 MB	448 KB
OS Architecture	ARMv8 (64-bit)	ARMv6Z (32-bit)	Xtensa LX6 (32-bit)
WiFi Modes	802.11ac	802.11n	802.11b/g/n

device experiments. These devices represent varying degrees of resource constraints, from multi-core processors to ultra-low-power microcontrollers. Importantly, all three devices support IEEE 802.11 wireless networking, which remains the dominant connectivity standard across consumer and industrial IoT deployments. These devices demonstrate a representative spectrum of real-world IoT hardware. Table 4 shows a comparison of the device capabilities.

IoT Cloud. We deploy a custom IoT cloud instance on an Ubuntu 20.04 server that mimics a typical vendor-managed cloud. This deployment consists of standard web components, including an NGINX reverse proxy, a Flask-based web server, and a uWSGI application backend (Figure 2). To support Atlas 's domain-bound identity model, we additionally configure an authoritative nameserver (using PowerDNS) responsible for managing the zone file associated with the vendor's root domain. The zone file maintains DNS resource records—including A, CNAME, and TXT entries—that map device UUIDs to device-specific subdomains, as described in §4.2.

In contrast to our setup, most commercial IoT cloud deployments rely on recursive DNS resolvers (*e.g.,* provided by AWS Route 53 or Cloudflare DNS) and do not maintain their own authoritative DNS infrastructure. This limits the ability to programmatically manage per-device subdomain records or tightly control DNS-based identity bindings. By operating an authoritative DNS server within our IoT cloud instance, Atlas enables dynamic and granular management of device-specific DNS entries.

For baseline comparisons, we also configure a deployment using AWS IoT Core, provisioning several Raspberry Pi 4 devices using the official AWS SDK[5].

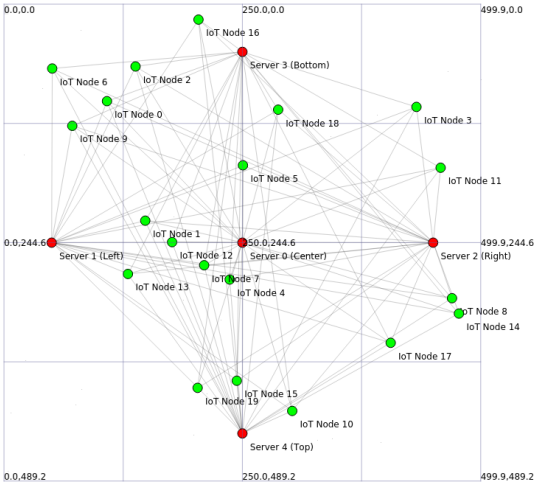


Fig. 9: Smart City Network Simulation: Mobile IoT Nodes (*e.g.,* autonomous vehicles) interacting with static Servers (*e.g.,* traffic light systems) for mobility management or emergency respor[illegible]

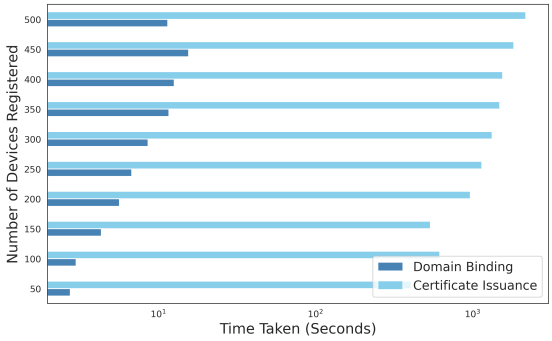


Fig. 10: Bulk registration of devices using Atlas is highly scalable.

AWS IoT provides out-of-the-box support for device provisioning, telemetry, and PKI services, serving as a representative model of cloud-mediated IoT infrastructure. This baseline allows us to contrast the operational and performance characteristics of traditional cloud-based device management with the Atlas architecture.

IoT Ecosystem. *Smart Home Testbed* — To demonstrate Atlas in realistic deployment scenarios, we develop two testbeds. First, a smart home testbed consisting of heterogeneous IoT devices (RPi4, RPi0W, ESP32) connected through a commodity home router and coordinated via a local automation hub (using an MQTT broker [25] on an RPi4). We contrast this with an AWS IoT-mediated setup, where two RPi4 devices, provisioned using AWS IoT Core, communicate through an IoT cloud (AWS IoT). We implemented cloud IoT applications using an AWS Lambda service that processes messages on the MQTT channels using AWS IoT Core.

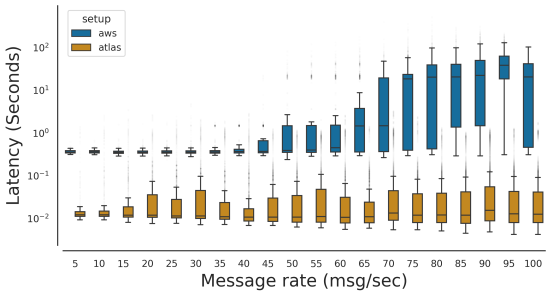


Fig. 11: Real-world smart home applications developed using the Atlas framework are consistently performant under scale workload and secure than the current setups that require mediation through IoT Clouds (AWS-IoT), which lead to high-variability in application performance.

Smart City Testbed — One of the biggest challenges when designing IoT solutions is validating their applicability over large-scale deployments. It is impractical to build large scale testbeds with actual devices. Additionally, most commercial IoT devices are closed source, thus, modifying them for specific use cases is not viable. In order to validate the applicability of the Atlas framework over large-scale deployments, such as smart cities, we leverage simulators such as ns-3 [46]. ns-3 is one of the most widely used discrete-event network simulator for Internet systems, targeted primarily for research and educational purposes. We program a smart city model that reflects a realistic scenario where mobile IoT nodes (*e.g.,* autonomous vehicles) have to periodically interact with static IoT gateways (*e.g.,* traffic lights) spread throughout the city. Figure 9 demonstrates a small scale simulation, for better visualization, where 15 mobile IoT nodes are traversing through a $500mX500m$ city grid using the *RandomWaypointMobilityModel* with speed varying from $5 - 25m/sec$. There are 5 statically allocated IoT gateways placed within the city which act as the aggregation nodes for device communications. Every mobile IoT node is programmed with a simple P2P application that looks for the nearest IoT gateway (within $100m$) every $2secs$, and starts sending a burst of messages to the gateway.

Under the Atlas framework, these messages are transmitted to the gateway directly using secure D2D communication. However, under the current status quo of IoT cloud setups, this connection would be transmitted via vendor-specific clouds. We model this behavior using a Weibull distribution (β=0.5, λ=6.37 s) fitted to our AWS IoT measurements (Figure 1b).