Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: October 10, 2026

R. Sharif
CyberSecAI Ltd

April 10, 2026

Agent Public Key Infrastructure (APKI): Certificate-Based Identity and Trust for Autonomous AI Agents

draft-sharif-apki-agent-pki-00

Status of This Memo

Copyright Notice

Abstract

Autonomous artificial intelligence (AI) agents are increasingly performing actions on the Internet that require verifiable identity: financial transactions, regulated data access, tool invocations, and inter-agent coordination. Traditional Public Key Infrastructure (PKI) based on X.509 certificates was designed for human-operated clients and long-lived servers. It lacks primitives for graduated trust scoring, capability constraints, delegation chains, model provenance, and the ephemeral lifecycles characteristic of AI agents.

This document defines Agent Public Key Infrastructure (APKI), a certificate-based identity and trust system for autonomous AI agents. APKI extends X.509v3 with five agent-specific extensions, defines the agent:// URI scheme for agent identification, specifies Agent Transparency Logs modelled on Certificate Transparency (RFC 9162), and provides mechanisms for cross-organizational trust federation. APKI is designed to be compatible with existing PKI deployments, SPIFFE workload identity, and the IETF WIMSE working group's specifications.

Table of Contents

1. Introduction

1.1. The Agent Identity Problem

Large language models (LLMs) and related machine learning systems have enabled the deployment of autonomous software agents that operate with minimal human supervision. These agents initiate financial transactions, query regulated databases, invoke external tools via protocols such as the Model Context Protocol (MCP), and coordinate with other agents in multi-agent workflows.

As of early 2026, the MCP ecosystem alone comprises over 97 million SDK downloads and 13,000 registered servers. According to the HID Global PKI Survey (March 2026), 16% of enterprises are already issuing digital certificates to AI agents, and 34% cite AI agent certificates as a top PKI trend.

Despite this rapid adoption, no standard exists for agent-specific certificate-based identity. Agents currently authenticate using mechanisms designed for human users (OAuth 2.0 bearer tokens, API keys) or generic workloads (SPIFFE SVIDs). These mechanisms lack the semantics required for agent trust decisions: graduated trust levels, capability constraints, delegation chains, and model provenance.

This document defines Agent Public Key Infrastructure (APKI), a certificate-based system that provides verifiable identity and graduated trust for autonomous AI agents.

1.2. Why Traditional PKI Is Insufficient

Traditional PKI based on X.509 certificates [RFC5280] exhibits the following deficiencies when applied to AI agents:

Temporal Mismatch: X.509 certificates have validity periods of days to years. AI agents may be created, execute a task, and terminate within milliseconds. Long-lived certificates create unnecessarily wide vulnerability windows for ephemeral entities.

Binary Trust: A certificate is either valid or revoked. Agents require graduated trust: a newly created agent from an unknown provider should receive lower trust than an established agent with a verified track record.

No Capabilities: Certificates assert identity but do not constrain actions. Agents need capabilities bound to their identity: which tools they may invoke, what spend limits apply, what data they may access.

No Delegation: When a human delegates authority to an agent that spawns sub-agents, X.509 provides no mechanism to represent this delegation chain with capability attenuation.

No Provenance: Certificates do not attest which AI model executes within an agent, what version it runs, or what framework hosts it.

Revocation Latency: CRLs refresh on hourly or daily intervals. For agents executing thousands of actions per minute, revocation latency measured in hours is unacceptable.

No Transparency: Certificate Transparency [RFC9162] covers TLS certificates but no equivalent exists for agent identity.

1.3. Design Goals

APKI is designed with the following goals:

1. Graduated trust embedded in certificates, not external systems
2. Short-lived certificates (minutes, not years) as the default
3. Capability constraints bound to agent identity
4. Cryptographic delegation chains with monotonic attenuation
5. Transparency logs for agent certificate issuance
6. Compatibility with existing PKI, SPIFFE, and WIMSE

7. Post-quantum algorithm readiness

8. Sub-second revocation capability

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Agent: An autonomous software entity, typically powered by a language model, that performs actions on the Internet with minimal human supervision.

Agent Certificate: An X.509v3 digital certificate issued to an agent, containing the agent's public key, an agent URI in the Subject Alternative Name, and one or more APKI-specific extensions.

Agent CA: A Certificate Authority within the APKI hierarchy that issues Agent Certificates. An Agent CA is typically an Organizational Intermediate CA.

Agent Root CA: The trust anchor of an APKI deployment. Operates offline and issues certificates only to Organizational Intermediate CAs.

Agent URI: A Uniform Resource Identifier following the agent:// scheme that uniquely identifies an agent within a trust domain.

Trust Score: A numerical value between 0 and 100 inclusive that represents an agent's current trust level. Embedded in the AgentTrustScore certificate extension.

Trust Tier: A classification derived from the trust score: UNTRUSTED (0-19), RESTRICTED (20-39), STANDARD (40-59), ELEVATED (60-79), FULL (80-100).

Trust Decay: The mechanism by which an agent's trust score decreases over time in the absence of positive behavioural signals.

Capability Constraint: A binding between an agent's identity and specific permitted operations, including tool URIs, spend limits, rate limits, and scope descriptors.

Delegation Chain: A cryptographically linked sequence of agent certificates representing authority delegation from a human principal through one or more agents.

Monotonic Attenuation: The invariant that each delegation in a chain MUST narrow (never expand) the delegated capabilities.

Agent Transparency Log: An append-only Merkle hash tree that records agent certificate issuance, enabling monitoring for unauthorized agent registrations.

Signed Agent Timestamp (SAT): A signed data structure returned by an Agent Transparency Log upon certificate submission, analogous to the Signed Certificate Timestamp (SCT) in Certificate Transparency.

Behavioural Attestation: Evidence that an agent's declared

capabilities have been verified by a specified method.

Model Provenance: Information about the AI model powering an agent, including model family, version, and hosting framework.

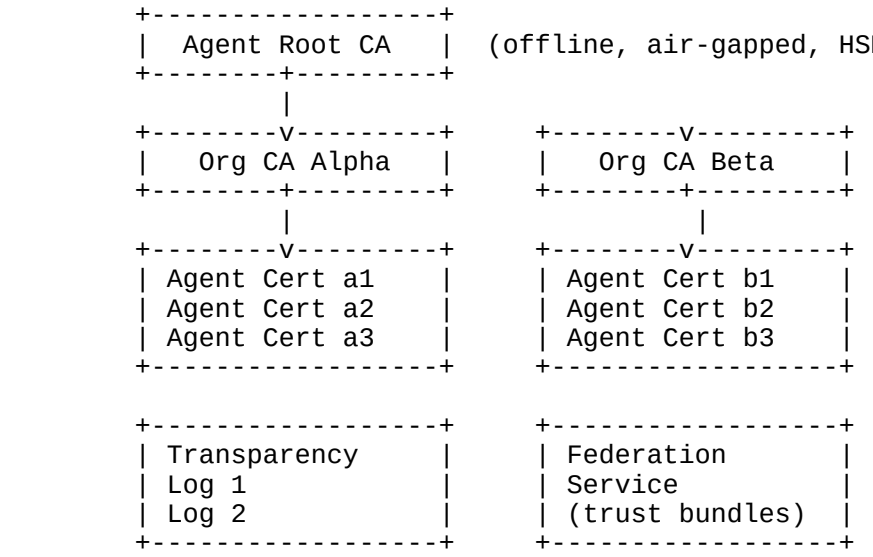
Federation: The mechanism by which organizations exchange trust bundles to enable cross-organizational agent interactions.

Trust Bundle: A set of CA certificates representing the trust anchors for an APKI trust domain, distributed for federation.

3. Architecture Overview

3.1. Component Hierarchy

The APKI system comprises the following components:



Agent Root CA: The trust anchor for the entire hierarchy. It operates offline with private key material in a Hardware Security Module (HSM). Issues certificates only to Organizational Intermediate CAs. Validity period: 10-20 years.

Organizational Intermediate CA: Each organization operating AI agents deploys one or more Intermediate CAs. These are online, automated systems that issue short-lived agent certificates. Validity period: 1-5 years.

Agent End-Entity Certificates: Short-lived certificates issued to individual agents. Default validity: 1 hour. Configurable range: 5 minutes to 24 hours.

Agent Transparency Logs: Independent append-only Merkle trees recording agent certificate issuance.

Federation Service: Distributes and consumes trust bundles for cross-organizational agent interactions.

3.2. Trust Model

APKI replaces the binary valid/revoked trust model of traditional PKI with a graduated trust model:

1. Every agent certificate contains a trust score (0-100)

2. Trust scores decay over time without positive signals

3. Relying parties enforce minimum trust score thresholds

4. Certificate re-issuance occurs when scores change materially

5. Trust decay serves as soft revocation -- agents that cease positive operations gradually lose access

This model is analogous to a credit score for financial transactions: the score reflects accumulated evidence about the agent's trustworthiness, not merely the validity of a credential.

4. Agent URI Scheme

4.1. Syntax

The agent URI scheme identifies agents within APKI certificates:

```
agent-uri = "agent://" trust-domain "/" org "/" type "/" instance

trust-domain = dns-name
org          = 1*( ALPHA / DIGIT / "-" / "_" )
type         = 1*( ALPHA / DIGIT / "-" / "_" )
instance     = 1*( ALPHA / DIGIT / "-" / "_" )
```

The trust-domain component MUST be a valid DNS name identifying the organizational trust boundary. The org component identifies the operating entity. The type component identifies the agent's functional classification. The instance component uniquely identifies the specific agent instance.

4.2. Examples

```
agent://cybersecai.co.uk/payments/payment-bot/a1b2c3d4
agent://bank.example.com/kyc/verification-agent/e5f6g7h8
agent://cloud.example.org/data/pipeline-agent/i9j0k1l2
```

4.3. Placement in X.509 Certificates

The agent URI MUST be placed in the X.509 Subject Alternative Name (SAN) extension as a uniformResourceIdentifier, as defined in Section 4.2.1.6 of [RFC5280]. This follows the same pattern established by SPIFFE for workload identity (spiffe:// URIs in SANs).

Implementations MUST support agent:// URIs in the SAN. Implementations SHOULD also support spiffe:// URIs in the same certificate for environments that use both APKI and SPIFFE.

5. Agent Certificate Profile

5.1. Base Certificate Fields

An APKI agent certificate is a standard X.509v3 certificate [RFC5280] with the following field requirements:

Version: MUST be v3 (value 2).

Serial Number: MUST be a unique positive integer assigned by the issuing CA, as per [RFC5280] Section 4.1.2.2.

Signature Algorithm: MUST be one of: ecdsa-with-SHA256 (for P-256 keys), Ed25519, or ML-DSA-65 (when post-quantum profiles are standardized).

Issuer: The distinguished name of the Organizational Intermediate CA.

Validity: The notBefore and notAfter fields. The validity period SHOULD be between 5 minutes and 24 hours. The RECOMMENDED default is 1 hour.

Subject: MAY be empty. Agent identity is carried in the SAN.

Subject Public Key Info: The agent's public key. MUST be one of: EC P-256, Ed25519, or ML-DSA-65.

Extensions: MUST include SubjectAlternativeName with the agent URI. MUST include at least one APKI extension. SHOULD include all five APKI extensions for maximum interoperability.

5.2. AgentTrustScore Extension

The AgentTrustScore extension embeds a graduated trust score within the certificate.

OID: id-pe-agentTrustScore (TBD)

Criticality: MUST be marked non-critical.

ASN.1 Definition:

```
AgentTrustScore ::= SEQUENCE {
    score               INTEGER (0..100),
    trustTier           ENUMERATED {
        untrusted           (0),
        restricted          (1),
        standard            (2),
        elevated            (3),
        full                (4)
    },
    decayRate           INTEGER (0..100),
    lastUpdated         GeneralizedTime,
    computationMethod   UTF8String OPTIONAL
}
```

The score field contains the agent's current trust score. The trustTier field contains the tier classification derived from the score. The decayRate field specifies the number of trust points lost per hour in the absence of positive signals. The lastUpdated field records when the score was last computed. The computationMethod field optionally identifies the scoring algorithm.

Relying parties MUST verify that the trust score meets their minimum threshold before granting access. A relying party requiring ELEVATED trust MUST reject certificates with scores below 60.

5.3. AgentCapabilities Extension

The AgentCapabilities extension binds specific operational capabilities to the agent's identity.

OID: id-pe-agentCapabilities (TBD)

Criticality: SHOULD be marked non-critical.

ASN.1 Definition:

```
AgentCapabilities ::= SEQUENCE {
    capabilities  SEQUENCE OF Capability
}

Capability ::= SEQUENCE {
    toolUri       IA5String,
    scope         UTF8String,
    spendLimit    SpendConstraint OPTIONAL,
    rateLimit     RateConstraint OPTIONAL
}

SpendConstraint ::= SEQUENCE {
    maxPerTransaction  INTEGER OPTIONAL,
    maxPerPeriod       INTEGER OPTIONAL,
    periodSeconds      INTEGER OPTIONAL,
    currency           PrintableString
}

RateConstraint ::= SEQUENCE {
    maxRequests    INTEGER,
    periodSeconds  INTEGER
}
```

The toolUri field identifies a permitted tool or API endpoint. The scope field describes the operational scope. The spendLimit field constrains monetary operations. Monetary values are expressed in minor currency units (e.g., pence, cents). The rateLimit field constrains invocation frequency.

5.4. AgentDelegation Extension

The AgentDelegation extension encodes the delegation chain from a human principal through one or more agents.

OID: id-pe-agentDelegation (TBD)

Criticality: SHOULD be marked non-critical.

ASN.1 Definition:

```
AgentDelegation ::= SEQUENCE {
    parentCertHash      OCTET STRING (SIZE(32)),
    delegationDepth     INTEGER (0..255),
    maxDelegationDepth  INTEGER (0..255),
    attenuationRules    AttenuationRules,
    humanPrincipal      UTF8String OPTIONAL
}

AttenuationRules ::= SEQUENCE {
    capabilitiesSubset  BOOLEAN DEFAULT TRUE,
    maxTrustScore       INTEGER (0..100) OPTIONAL,
    maxSpendLimit       INTEGER OPTIONAL,
    scopeNarrowing      UTF8String OPTIONAL
}
```

The parentCertHash field MUST contain the SHA-256 hash of the parent agent’s DER-encoded certificate, creating a cryptographic binding. The delegationDepth field records the current position in the chain (0 for agents directly delegated by humans). The

maxDelegationDepth field sets the upper bound. The RECOMMENDED maximum delegation depth is 5.

The fundamental invariant is monotonic attenuation: a child agent's capabilities MUST be a strict subset of its parent's capabilities. The issuing CA MUST enforce this invariant at certificate issuance time. A child agent MUST NOT have a higher trust score, greater spend limit, or broader scope than its parent.

5.5. AgentProvenance Extension

The AgentProvenance extension attests the computational provenance of the agent.

OID: id-pe-agentProvenance (TBD)

Criticality: MUST be marked non-critical.

ASN.1 Definition:

```
AgentProvenance ::= SEQUENCE {
    modelFamily     UTF8String,
    modelVersion    UTF8String,
    framework       UTF8String,
    organizationId  UTF8String,
    buildHash       OCTET STRING (SIZE(32)) OPTIONAL,
    attestEvidence  OCTET STRING OPTIONAL
}
```

The modelFamily field identifies the AI model family (e.g., "claude", "gpt", "gemini"). The modelVersion field identifies the specific version. The framework field identifies the hosting framework (e.g., "langchain", "crewai", "mcp-sdk"). The organizationId field identifies the operating organization. The buildHash field optionally contains the SHA-256 hash of the agent's source code or binary.

5.6. AgentBehaviouralAttestation Extension

The AgentBehaviouralAttestation extension links declared capabilities to verifiable attestation evidence.

OID: id-pe-agentBehaviouralAttestation (TBD)

Criticality: MUST be marked non-critical.

ASN.1 Definition:

```
AgentBehaviouralAttestation ::= SEQUENCE {
    declaredCapabilitiesHash  OCTET STRING (SIZE(32)),
    attestationMethod         ENUMERATED {
        selfDeclared    (0),
        caVerified      (1),
        thirdParty      (2),
        hardwareBound   (3)
    },
    attestorIdentity  UTF8String OPTIONAL,
    attestationTime   GeneralizedTime,
    evidenceUri       IA5String OPTIONAL
}
```

Attestation methods provide increasing levels of assurance: self-declared provides minimal assurance; CA-verified indicates

the issuing CA tested the agent; third-party indicates an independent auditor verified capabilities; hardware-bound indicates capabilities are enforced by a trusted execution environment.

6. Trust Score Semantics

6.1. Trust Tiers

Trust scores map to five tiers:

Score	Tier	Access Level
0-19	UNTRUSTED	No access
20-39	RESTRICTED	Read-only, sandboxed
40-59	STANDARD	Normal operations within scope
60-79	ELEVATED	Sensitive operations, regulated
80-100	FULL	Financial, production, audited

Relying parties MUST define their minimum required trust tier. An agent certificate with a trust score below the minimum MUST be rejected.

6.2. Trust Decay

Trust scores decay over time without positive behavioural signals. The decay rate is specified in the AgentTrustScore extension as points lost per hour.

The decay formula is:

```
current_score = max(0, last_score - (decay_rate * hours_elapsed))
```

Where hours_elapsed is calculated from the lastUpdated timestamp in the AgentTrustScore extension to the current time.

A typical decay rate of 2 points per hour means an agent with an initial score of 80 reaches STANDARD threshold (60) after 10 hours and RESTRICTED threshold (40) after 20 hours without positive interaction.

Relying parties SHOULD compute the decayed score when evaluating certificates, not rely solely on the score at issuance time.

6.3. Trust Updates

Relying parties report agent behaviour to the Trust Score Computation Engine at the issuing CA:

```
Successful operation:     +0.5 to +2 points
Failed operation:         -1 to -5 points
Policy violation:         -10 to -25 points
Anomalous behaviour:      -5 to -20 points
Security incident:        immediate reset to 0
```

The reporting protocol is out of scope for this document and will be defined in a companion specification.

6.4. Re-issuance Triggers

When the trust score changes by more than a configurable threshold (RECOMMENDED default: 10 points), the CA SHOULD issue a replacement certificate with the updated score. The previous certificate remains valid until natural expiry.

When the trust score falls below a configurable minimum (RECOMMENDED default: 20), the CA SHOULD revoke the certificate.

7. Agent Certificate Lifecycle

7.1. Enrollment

Agent enrollment follows an ACME-like [RFC8555] pattern adapted for attestation-based validation:

1. The agent generates an ECDSA P-256 or Ed25519 key pair.
2. The agent constructs a Certificate Signing Request (CSR) per [RFC2986] containing the desired agent URI and attestation evidence (model provenance, framework identity, organizational binding).
3. The agent submits the CSR to its Organizational CA via a TLS-protected channel.
4. The CA verifies attestation evidence against its policies.
5. The Trust Score Computation Engine calculates the initial score based on: attestation strength (0-30 points), organization reputation (0-20 points), model provenance (0-15 points), code attestation (0-15 points), and historical data (0-20 points).
6. The CA constructs the X.509v3 certificate with APKI extensions populated.
7. The CA submits the certificate to at least one Agent Transparency Log.
8. The log returns a Signed Agent Timestamp (SAT).
9. The SAT is embedded in the certificate and delivered to the agent.

7.2. Issuance

The CA MUST verify the following before issuance:

1. The agent URI trust-domain matches the CA’s authority
2. Attestation evidence is valid per the CA’s policies
3. If a delegation extension is present, the parent certificate is valid and the monotonic attenuation invariant holds
4. The certificate has been submitted to at least one Transparency Log and a valid SAT has been received

7.3. Renewal

Automated renewal SHOULD occur at 50% of the validity period. At renewal:

1. The agent presents its current certificate and fresh attestation evidence

2. The CA recalculates the trust score incorporating all behavioural signals received since last issuance

3. A new certificate is issued with the updated trust score

4. The new certificate is submitted to Transparency Logs

7.4. Revocation

APKI provides three complementary revocation mechanisms:

Natural Expiry: Agent certificates have short validity periods (default 1 hour). A compromised certificate expires naturally, limiting exposure. This is the primary mechanism.

Certificate Revocation Lists: The CA publishes CRLs with a refresh interval of 60 seconds or less. CRLs include cascade-revoked certificates from delegation chains.

OCSP: The CA provides OCSP responders with sub-second response times. OCSP stapling is supported.

Trust Decay: Serves as soft revocation. An agent that ceases operations sees its score decay below thresholds, losing access without explicit revocation.

8. Agent Transparency Logs

8.1. Log Structure

Agent Transparency Logs are append-only Merkle hash trees following the architecture of Certificate Transparency [RFC9162] applied to agent certificates.

Each leaf in the Merkle tree is the SHA-256 hash of a submitted agent certificate (DER-encoded). The log operator periodically signs the tree head (Signed Tree Head, STH), enabling clients to verify log consistency and completeness.

Log operators MUST be independent of the CA hierarchy. Multiple independent logs SHOULD be operated to prevent single points of trust.

8.2. Signed Agent Timestamps

When a CA submits a certificate to a log, the log returns a Signed Agent Timestamp (SAT) comprising:

```
SAT ::= SEQUENCE {
    version        INTEGER,
    logId          OCTET STRING (SIZE(32)),
    timestamp      INTEGER,
    certHash       OCTET STRING (SIZE(32)),
    signature      OCTET STRING
}
```

The logId is the SHA-256 hash of the log’s public key. The timestamp is milliseconds since the Unix epoch. The certHash is the SHA-256 hash of the submitted certificate. The signature is computed over all preceding fields using the log’s private

key.

CAs MUST embed at least one SAT in the issued certificate via a SCTList-like extension, analogous to [RFC6962] Section 3.3.

8.3. Monitoring

Organizations SHOULD monitor Agent Transparency Logs for:

1. Certificates issued within their trust domain without authorization
2. Agents claiming organizational affiliation without proper delegation
3. Anomalous patterns such as bulk registration from unknown CAs
4. Agents with trust scores exceeding organizational policy

9. Delegation Chains

9.1. Monotonic Attenuation Invariant

The fundamental invariant of APKI delegation is monotonic attenuation: at each level of delegation, the child agent's authority MUST NOT exceed the parent's. Specifically:

1. The child's capability set MUST be a subset of the parent's
2. The child's trust score MUST NOT exceed the parent's
3. The child's spend limits MUST NOT exceed the parent's
4. The child's scope MUST be equal to or narrower than the parent's
5. The child's delegation depth MUST be parent's depth plus one

The issuing CA MUST enforce this invariant. Relying parties SHOULD verify it by walking the delegation chain.

9.2. Delegation Issuance

When parent Agent A requests delegation to child Agent B:

1. Agent A presents its certificate and the requested child capabilities to the CA.
2. The CA verifies Agent A's certificate is valid and its trust score meets a minimum delegation threshold.
3. The CA verifies the requested child capabilities satisfy the monotonic attenuation invariant.
4. The CA verifies the delegation depth would not exceed maxDelegationDepth.
5. The CA issues Agent B's certificate with the AgentDelegation extension containing Agent A's certificate hash.

9.3. Cascade Revocation

When a parent agent's certificate is revoked, all descendant certificates in the delegation chain MUST be automatically revoked. The CA maintains a delegation tree index and traverses it upon revocation. OCSP responses for descendant certificates MUST immediately return "revoked" status.

10. Federation

10.1. Trust Bundle Exchange

Organizations publish APKI trust bundles at a well-known URI:

```
https://{trust-domain}/.well-known/apki-trust-bundle
```

A trust bundle is a JSON document containing:

```
{
    "trust_domain": "cybersecai.co.uk",
    "version": 1,
    "issued_at": "2026-04-10T12:00:00Z",
    "expires_at": "2026-04-11T12:00:00Z",
    "ca_certificates": [
        "<base64-encoded DER certificate>"
    ],
    "minimum_trust_score": 40,
    "supported_algorithms": ["ES256", "Ed25519"]
}
```

Trust bundles MUST be served over HTTPS. Trust bundles SHOULD be refreshed at least daily. This format is compatible with SPIFFE trust bundle federation.

10.2. Federation Verification

When Organization A receives a request from an agent certified by Organization B's CA:

1. Organization A retrieves Organization B's trust bundle (cached with configurable TTL).
2. Organization A verifies the agent's certificate chain against Organization B's trust bundle.
3. Organization A checks the agent's trust score against its minimum threshold for federated agents.
4. If all checks pass, access is granted per Organization A's federation policies.

10.3. Trust Score Mapping

Organizations MAY define trust score mapping functions to translate between different scoring criteria. For example, Organization A may consider Organization B's agents to be 10 points less trustworthy due to lower attestation requirements.

The mapping function is applied after retrieving the certificate trust score and before comparing against the local minimum threshold.

11. Security Considerations

11.1. Threat Model

This section defines the threat model for APKI. The analysis follows the methodology of [RFC3552] and considers adversaries at multiple layers of the APKI architecture.

11.1.1. Attacker Capabilities

APKI considers the following attacker classes:

Network Attacker (Dolev-Yao): An attacker who can observe, intercept, modify, and inject messages on any network path. This is the standard Internet threat model. APKI mitigates this through TLS for all communications and cryptographic signatures on all certificates, timestamps, and trust bundles.

Compromised Agent: An agent whose private key has been extracted or whose runtime has been compromised (e.g., via prompt injection, model manipulation, or host compromise). APKI mitigates this through short-lived certificates, trust decay, and cascade revocation.

Malicious Operator: An organization that operates agents with the intent to deceive relying parties about the agents' capabilities, trust level, or provenance. APKI mitigates this through Transparency Logs, third-party attestation, and federated trust with minimum score thresholds.

Compromised CA: An Organizational Intermediate CA whose private key has been compromised, enabling issuance of arbitrary agent certificates. APKI mitigates this through Transparency Logs (detection), Root CA revocation (response), and multi-log consistency checks (prevention of silent mis-issuance).

Insider Attacker: A human with legitimate access to the CA or Trust Score Computation Engine who abuses their position to inflate scores, issue unauthorized certificates, or suppress revocations. APKI mitigates this through separation of duties (CA operator vs. score computation), Transparency Log audit trails, and the requirement for multiple independent log operators.

11.1.2. Assets Under Protection

The following assets are protected by APKI:

Agent Identity: The binding between an agent's public key and its agent URI. Compromise enables impersonation.

Trust Score Integrity: The accuracy of trust scores embedded in certificates. Manipulation enables privilege escalation.

Delegation Chain Integrity: The correctness of parent-child relationships and the monotonic attenuation invariant. Compromise enables unauthorized capability expansion.

Capability Constraints: The binding between an agent's identity and its permitted operations. Bypass enables unauthorized actions (financial transactions, data access, tool invocation).

Transparency Log Integrity: The append-only property of logs and the completeness of log entries. Compromise enables undetected mis-issuance.

Model Provenance: The accuracy of claims about which AI model powers an agent. Falsification enables compliance violations

in regulated industries.

11.1.3. Out of Scope

The following threats are explicitly out of scope for APKI:

1. Compromise of the offline Root CA (assumed to be physically secured in an air-gapped HSM environment)

2. Vulnerabilities in the underlying AI model (prompt injection, jailbreaking, hallucination) -- APKI provides identity and trust, not model safety

3. Side-channel attacks on cryptographic implementations

4. Compromise of all Transparency Log operators simultaneously

5. Quantum computing attacks on classical algorithms (addressed by post-quantum migration path in Section 11.3)

11.2. Threat Analysis

11.2.1. T1: Trust Score Inflation

Threat: An attacker fabricates positive behavioural signals to inflate an agent's trust score, causing the CA to issue certificates with artificially high scores.

Impact: HIGH. An agent with an inflated score gains access to operations requiring ELEVATED or FULL trust (financial transactions, regulated data, production environments).

Likelihood: MODERATE. Requires the ability to submit authenticated behavioural signals to the Trust Score Computation Engine.

Mitigations:

- Behavioural signals MUST be authenticated using the reporting relying party's credentials
- The Trust Score Computation Engine MUST rate-limit signal acceptance (RECOMMENDED: maximum 10 signals per agent per hour from any single relying party)
- Anomaly detection SHOULD flag sudden score increases (e.g., more than 20 points in 24 hours)
- Trust scores SHOULD incorporate multiple independent signal sources; no single relying party should be able to move a score by more than 15 points
- Historical signal patterns SHOULD be compared against population baselines

11.2.2. T2: Trust Score Gaming via Decay Reset

Threat: An attacker allows an agent's score to decay, then generates a burst of positive signals to obtain a fresh certificate with a higher-than-warranted score.

Impact: MODERATE. The reset score is bounded by the CA's computation algorithm and cannot exceed the maximum initial score for the agent's attestation level.

Mitigations:

- The CA MUST NOT issue a certificate with a score higher than the agent's historical peak minus any penalty history
- Decay history MUST be maintained across certificate renewals
- The decay rate SHOULD increase for agents with a history

of rapid score fluctuations

11.2.3. T3: Delegation Chain Privilege Escalation

Threat: An attacker creates a delegation chain that bypasses the monotonic attenuation invariant, resulting in a child agent with broader capabilities than its parent.

Impact: CRITICAL. The child agent could execute operations (e.g., financial transactions, data exfiltration) that the parent was not authorized to delegate.

Likelihood: LOW if the CA correctly implements attenuation checks. Higher if the CA has implementation bugs.

Mitigations:

- The CA MUST verify the monotonic attenuation invariant before issuing any delegated certificate
- Relying parties SHOULD independently verify the invariant by walking the delegation chain (checking each parent certificate's capabilities are a superset of the child's)
- The delegation depth MUST be checked against maxDelegationDepth at issuance
- Implementations MUST treat attenuation verification failures as CRITICAL errors and refuse issuance

11.2.4. T4: Rogue Certificate Issuance

Threat: A compromised Organizational Intermediate CA issues agent certificates with fabricated trust scores, false capabilities, or spoofed agent URIs.

Impact: CRITICAL within the compromised trust domain. Agents with fraudulent certificates could impersonate legitimate agents, execute unauthorized operations, and evade detection.

Likelihood: LOW (requires CA key compromise or insider attack).

Mitigations:

- All certificates MUST be submitted to at least one Transparency Log before issuance; certificates without valid SATs MUST be rejected by relying parties
- Organizations MUST monitor Transparency Logs for certificates issued within their trust domain
- The Root CA MUST maintain the ability to revoke any Intermediate CA certificate within 60 seconds
- Cross-log consistency checks (gossip protocols) SHOULD detect split-view attacks where a rogue CA submits different certificates to different logs
- Relying parties SHOULD require SATs from at least two independent logs for ELEVATED and FULL trust operations

11.2.5. T5: Transparency Log Manipulation

Threat: An attacker compromises a Transparency Log operator and inserts, modifies, or suppresses log entries.

Impact: HIGH. Enables undetected rogue certificate issuance (T4) or concealment of compromised agents.

Likelihood: LOW if multiple independent logs are required.

Mitigations:

- Log operators MUST sign tree heads at regular intervals (RECOMMENDED: every 60 seconds)
- Monitors MUST verify log consistency by requesting

inclusion proofs and consistency proofs
- Relying parties SHOULD accept SATs only from logs that have been independently audited within the last 24 hours
- A minimum of two independent logs operated by different entities in different jurisdictions is RECOMMENDED
- Log operators SHOULD implement append-only storage with hardware write-once guarantees where available

11.2.6. T6: Agent Impersonation via Key Theft

Threat: An attacker extracts an agent's private key and uses it to impersonate the agent during the certificate's validity period.

Impact: HIGH. The attacker can perform any operation the legitimate agent is authorized for.

Likelihood: MODERATE for software-stored keys. LOW for HSM-bound keys.

Mitigations:
- Short-lived certificates (default 1 hour) limit the window of exposure
- HSM-bound keys (PKCS#11, TPM, cloud KMS) prevent key extraction; implementations SHOULD default to HSM storage for agents with ELEVATED or FULL trust
- Trust decay reduces the value of stolen credentials over time even within the validity period
- Challenge-response protocols at the application layer provide additional proof-of-possession beyond certificate presentation
- Relying parties MAY require DPoP (Demonstration of Proof-of-Possession) for high-value operations

11.2.7. T7: Trust Domain Spoofing

Threat: An attacker registers a trust domain similar to a legitimate organization (e.g., cybersecai.co.uk vs. cybersec-ai.co.uk) and issues agent certificates that appear to belong to the legitimate organization.

Impact: HIGH. Relying parties may trust spoofed agents, leading to data exposure or financial loss.

Likelihood: MODERATE. Requires registering a similar domain and establishing an APKI CA, but does not require compromising any legitimate infrastructure.

Mitigations:
- Trust bundles MUST be exchanged via pre-configured, authenticated channels (not auto-discovered)
- Federation agreements MUST be established out-of-band with identity verification of the counterparty
- Relying parties MUST NOT accept agent certificates from trust domains not present in their configured trust bundle set
- Transparency Log monitors SHOULD alert on new trust domains that are typographically similar to existing monitored domains

11.2.8. T8: Capability Constraint Bypass

Threat: An agent or its operator attempts to invoke operations beyond those specified in the AgentCapabilities extension, hoping that the relying party does not enforce capability checks.

Impact: HIGH. Unauthorized financial transactions, data access, or tool invocations.

Likelihood: HIGH if relying parties treat capability checking as optional.

Mitigations:

- Relying parties MUST verify the AgentCapabilities extension before granting access to any protected resource
- The toolUri in the capability MUST match the resource being accessed; wildcard matching MUST NOT be used
- Spend limits MUST be enforced by the relying party (payment processor, API gateway) independently of the agent's self-reported spending
- Rate limits MUST be enforced server-side using the certificate's rate constraint values
- Implementations MUST fail closed: if the AgentCapabilities extension is absent or unparseable, access MUST be denied

11.2.9. T9: Model Provenance Falsification

Threat: An operator falsifies the AgentProvenance extension, claiming an agent uses an approved model (e.g., Claude) when it actually uses an unapproved or fine-tuned model.

Impact: MODERATE to HIGH in regulated industries where model approval is required for compliance.

Likelihood: MODERATE. Self-declared provenance is trivially falsifiable. CA-verified provenance is harder to fake.

Mitigations:

- Relying parties requiring high assurance MUST require attestationMethod of caVerified (1) or higher
- CAs providing caVerified attestation MUST verify model provenance through technical means (API introspection, model fingerprinting, or framework-level attestation)
- Third-party attestation (2) SHOULD be required for regulated use cases (financial services, healthcare)
- Hardware-bound attestation (3) using TEE attestation reports provides the highest assurance and SHOULD be used where available
- Build hashes in the AgentProvenance extension SHOULD be verified against a known-good build registry maintained by the model provider

11.2.10. T10: Cascade Revocation Evasion

Threat: A child agent detects that its parent is about to be revoked and races to perform unauthorized operations before the cascade revocation propagates.

Impact: MODERATE. Limited by the speed of cascade propagation and the short validity period of certificates.

Likelihood: LOW. Requires the child agent to have advance knowledge of the parent's revocation.

Mitigations:

- Cascade revocation MUST propagate to OCSP responders within 1 second of the parent's revocation
- CRLs MUST be updated within 60 seconds
- Relying parties performing high-value operations SHOULD perform real-time OCSP checks rather than relying on cached CRL data

- The delegation tree index maintained by the CA MUST be updated atomically with the parent's revocation

11.3. Post-Quantum Considerations

APKI certificates MUST support migration to post-quantum algorithms as standardized by NIST. The following migration path is RECOMMENDED:

Phase 1 (Current): ECDSA P-256 and Ed25519 as primary algorithms.

Phase 2 (Transition): Composite signatures combining a classical algorithm (ECDSA P-256 or Ed25519) with a post-quantum algorithm (ML-DSA-65) in the same certificate. Both signatures MUST be valid for the certificate to be accepted.

Phase 3 (Post-Quantum): ML-DSA-65 or SLH-DSA as standalone algorithms once classical algorithms are deprecated.

Implementations MUST support algorithm agility in all protocol messages to enable migration without infrastructure replacement. The Signature Algorithm field in agent certificates MUST be checked by relying parties against a locally configured list of acceptable algorithms.

11.4. Privacy Considerations

Agent URIs may reveal organizational structure, agent types, and deployment topology. The following privacy measures are RECOMMENDED:

1. Organizations SHOULD use opaque instance identifiers rather than sequential or predictable values

2. The type component of agent URIs SHOULD use generic functional names rather than names that reveal business logic

3. Transparency Log entries are public; organizations MUST consider that agent registrations, capabilities, and trust scores in Transparency Logs are observable by any party

4. Trust score values in certificates may reveal information about an agent's operational history; organizations operating in sensitive contexts SHOULD evaluate whether this information leakage is acceptable

5. Federation trust bundles reveal the existence of inter-organizational relationships; organizations SHOULD consider whether publishing trust bundles at well-known URIs exposes sensitive partnership information

11.5. Denial of Service Considerations

The following components are susceptible to denial of service:

Transparency Logs: Volumetric attacks against log submission endpoints can prevent certificate issuance. Log operators MUST implement rate limiting, geographic distribution, and DDoS mitigation.

OCSP Responders: Overwhelming OCSP responders can prevent revocation checking. OCSP stapling SHOULD be used to reduce responder load. Relying parties MUST define a fail-closed

policy when OCSP is unreachable.

Trust Score Computation Engine: Flooding the engine with behavioural signals can cause score instability. The engine MUST implement per-source rate limiting and signal authentication.

Federation Endpoints: Trust bundle endpoints may be targeted to prevent cross-organizational trust establishment. Trust bundles SHOULD be cached with configurable TTLs and served from CDN infrastructure.

12. IANA Considerations

This document requests the following IANA registrations:

12.1. URI Scheme Registration

Scheme name: agent

Status: Permanent

Applications/protocols: Agent Public Key Infrastructure (APKI)

Contact: IETF

Change controller: IETF

Reference: This document, Section 4

12.2. SMI Security for PKIX Certificate Extension OIDs

The following OIDs are requested under id-pe (1.3.6.1.5.5.7.1):

id-pe-agentTrustScore	(TBD1)
id-pe-agentCapabilities	(TBD2)
id-pe-agentDelegation	(TBD3)
id-pe-agentProvenance	(TBD4)
id-pe-agentBehaviouralAttestation	(TBD5)

12.3. Well-Known URI Registration

URI suffix: apki-trust-bundle

Change controller: IETF

Specification document: This document, Section 10.1

Related information: None

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017.

[RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021.

13.2. Informative References

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020.

[SPIFFE] SPIFFE Project, "Secure Production Identity Framework for Everyone (SPIFFE)", https://spiffe.io/.

[I-D.ietf-wimse-arch] WIMSE Working Group, "Workload Identity in Multi-System Environments Architecture", Internet-Draft draft-ietf-wimse-arch, 2024.

[I-D.niyikiza-oauth-attenuating-agent-tokens] Niyikiza, E., "Attenuating Authorization Tokens for Autonomous AI Agents", Internet-Draft draft-niyikiza-oauth-attenuating-agent-tokens-00, 2026.

[I-D.sharif-mcps-secure-mcp] Sharif, R., "MCPS: Cryptographic Security Layer for the Model Context Protocol", Internet-Draft draft-sharif-mcps-secure-mcp, 2026.

[I-D.sharif-agent-identity-framework] Sharif, R., "Agent Identity Framework: Trust and Identity for Autonomous AI Agents", Internet-Draft draft-sharif-agent-identity-framework-00, 2026.

[I-D.sharif-agent-payment-trust] Sharif, R., "Agent Payment Trust: Dynamic Financial Authorisation for Autonomous AI Agents", Internet-Draft draft-sharif-agent-payment-trust, 2026.

Appendix A. ASN.1 Module

```
   APKI-2026
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) id-mod(0)
         id-mod-apki-2026(TBD) }

   DEFINITIONS IMPLICIT TAGS ::=

   BEGIN

   IMPORTS
       Extensions
           FROM PKIX1Explicit88
               { iso(1) identified-organization(3) dod(6)
                 internet(1) security(5) mechanisms(5) pkix(7)
                 id-mod(0) id-pkix1-explicit(18) }
       ;

   -- OID Arc for APKI Extensions

   id-pe-agentTrustScore OBJECT IDENTIFIER ::=
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) pe(1) TBD1 }

   id-pe-agentCapabilities OBJECT IDENTIFIER ::=
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) pe(1) TBD2 }

   id-pe-agentDelegation OBJECT IDENTIFIER ::=
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) pe(1) TBD3 }

   id-pe-agentProvenance OBJECT IDENTIFIER ::=
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) pe(1) TBD4 }

   id-pe-agentBehaviouralAttestation OBJECT IDENTIFIER ::=
       { iso(1) identified-organization(3) dod(6) internet(1)
         security(5) mechanisms(5) pkix(7) pe(1) TBD5 }


   -- AgentTrustScore Extension

   AgentTrustScore ::= SEQUENCE {
       score               INTEGER (0..100),
       trustTier           ENUMERATED {
           untrusted           (0),
           restricted          (1),
           standard            (2),
           elevated            (3),
           full                (4)
       },
       decayRate           INTEGER (0..100),
       lastUpdated         GeneralizedTime,
       computationMethod   UTF8String OPTIONAL
   }


   -- AgentCapabilities Extension

   AgentCapabilities ::= SEQUENCE {
       capabilities  SEQUENCE OF Capability
   }

   Capability ::= SEQUENCE {
```

```
    toolUri       IA5String,
    scope         UTF8String,
    spendLimit    SpendConstraint OPTIONAL,
    rateLimit     RateConstraint OPTIONAL
}

SpendConstraint ::= SEQUENCE {
    maxPerTransaction  INTEGER OPTIONAL,
    maxPerPeriod       INTEGER OPTIONAL,
    periodSeconds      INTEGER OPTIONAL,
    currency           PrintableString
}

RateConstraint ::= SEQUENCE {
    maxRequests    INTEGER,
    periodSeconds  INTEGER
}


-- AgentDelegation Extension

AgentDelegation ::= SEQUENCE {
    parentCertHash      OCTET STRING (SIZE(32)),
    delegationDepth     INTEGER (0..255),
    maxDelegationDepth  INTEGER (0..255),
    attenuationRules    AttenuationRules,
    humanPrincipal      UTF8String OPTIONAL
}

AttenuationRules ::= SEQUENCE {
    capabilitiesSubset  BOOLEAN DEFAULT TRUE,
    maxTrustScore       INTEGER (0..100) OPTIONAL,
    maxSpendLimit       INTEGER OPTIONAL,
    scopeNarrowing      UTF8String OPTIONAL
}


-- AgentProvenance Extension

AgentProvenance ::= SEQUENCE {
    modelFamily     UTF8String,
    modelVersion    UTF8String,
    framework       UTF8String,
    organizationId  UTF8String,
    buildHash       OCTET STRING (SIZE(32)) OPTIONAL,
    attestEvidence  OCTET STRING OPTIONAL
}


-- AgentBehaviouralAttestation Extension

AgentBehaviouralAttestation ::= SEQUENCE {
    declaredCapabilitiesHash  OCTET STRING (SIZE(32)),
    attestationMethod         ENUMERATED {
        selfDeclared    (0),
        caVerified      (1),
        thirdParty      (2),
        hardwareBound   (3)
    },
    attestorIdentity  UTF8String OPTIONAL,
    attestationTime   GeneralizedTime,
    evidenceUri       IA5String OPTIONAL
}


-- Signed Agent Timestamp
```

```
SignedAgentTimestamp ::= SEQUENCE {
    version    INTEGER,
    logId      OCTET STRING (SIZE(32)),
    timestamp  INTEGER,
    certHash   OCTET STRING (SIZE(32)),
    signature  OCTET STRING
}

END
```

Appendix B. Example Agent Certificate

The following is a conceptual representation of an APKI agent certificate (actual DER encoding omitted for readability):

```
Certificate:
    Version: 3
    Serial Number: 7a:3b:c4:d5:e6:f7:08:19
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: CN=CyberSecAI Org CA, O=CyberSecAI Ltd, C=GB
    Validity:
        Not Before: Apr 10 12:00:00 2026 UTC
        Not After:  Apr 10 13:00:00 2026 UTC
    Subject: (empty)
    Subject Public Key Info:
        Algorithm: EC (P-256)
        Public Key: 04:a1:b2:c3:...
    X509v3 Extensions:
        Subject Alternative Name:
            URI: agent://cybersecai.co.uk/payments/bot/a1b2c3
        AgentTrustScore:
            Score: 75
            Tier: ELEVATED
            Decay Rate: 2 points/hour
            Last Updated: Apr 10 12:00:00 2026 UTC
        AgentCapabilities:
            Tool: mcp://stripe.com/charges/create
                Scope: payments
                Max Per Transaction: 100000 (GBP 1000.00)
                Max Per Period: 500000 (GBP 5000.00/day)
                Rate: 60 requests/3600 seconds
            Tool: mcp://sanctions.agentpass.co.uk/screen
                Scope: aml-screening
                Rate: 120 requests/3600 seconds
        AgentDelegation:
            Parent Cert Hash: sha256:8f4e2d...
            Delegation Depth: 1
            Max Delegation Depth: 3
            Attenuation: capabilities-subset=TRUE
            Human Principal: raza.sharif@cybersecai.co.uk
        AgentProvenance:
            Model: claude / opus-4-6
            Framework: mcp-sdk
            Organization: CyberSecAI Ltd
            Build Hash: sha256:3c7b9a...
        AgentBehaviouralAttestation:
            Capabilities Hash: sha256:5d1e8f...
            Method: caVerified
            Attestor: CyberSecAI Org CA
            Time: Apr 10 11:55:00 2026 UTC
        Signed Agent Timestamps:
            Log: CyberSecAI-ATL-01
            Time: Apr 10 11:59:58 2026 UTC
            Signature: 30:45:02:21:...
```

```
      Signature Algorithm: ecdsa-with-SHA256
      Signature Value: 30:46:02:21:...
```

Appendix C. Comparison with Existing Systems

Feature	X.509	SPIFFE	WIMSE	OAuth	DID	APKI
Cert-based	Yes	Yes	No	No	No	Yes
Trust scoring	No	No	No	No	No	Yes
Capabilities	No	No	No	Scope	VC	Yes
Delegation	Proxy	No	No	Chain	No	Yes
Provenance	No	No	No	No	No	Yes
Transparency	CT	No	No	No	No	ATL
Short-lived	Rare	Yes	N/A	Yes	N/A	Yes
Agent-specific	No	No	No	No	No	Yes
Federation	Cross	Yes	Yes	No	Yes	Yes

APKI is the first system to combine certificate-based identity with graduated trust scoring, capability constraints, delegation chains, model provenance, and transparency logging specifically designed for autonomous AI agents.

Author’s Address

Raza Sharif
CyberSecAI Ltd
London
United Kingdom

Email: raza.sharif@outlook.com
URI: https://cybersecai.co.uk