Lux ID: Decentralized Identity and Universal Access Management

A Five-Year Journey from Web2 to Web3 Identity

Lux Foundation
research@lux.network

Identity Working Group
id@lux.network

v2020.10 — October 2020 (Initial Version)
v2025.10 — October 2025 (Current Revision)

Reflecting 5 years of production deployment and continuous evolution

Abstract

We present Lux ID, a comprehensive decentralized identity (DID) and Identity Access Management (IAM) system that bridges Web2 and Web3 authentication paradigms. Initially deployed in October 2020, Lux ID has evolved over five years to support multi-protocol authentication (OAuth 2.0, OpenID Connect, SAML, CAS, LDAP), advanced security features (WebAuthn, TOTP, MFA), and blockchain-native identity via the `did:lux:address` format. The October 2025 revision introduces post-quantum cryptographic credentials, hardware security key enhancements, and seamless integration with the Lux blockchain ecosystem. This paper provides the complete DID specification, architectural details, security analysis, performance benchmarks, and comparisons with existing identity systems. Lux ID demonstrates that production-grade decentralized identity can coexist with traditional IAM requirements while maintaining superior security, privacy, and user experience.

Keywords: Decentralized Identity, DID, IAM, OAuth 2.0, WebAuthn, Post-Quantum Cryptography, Blockchain Identity, Multi-Protocol Authentication

Version History

- **v2020.10** (October 2020): Initial implementation based on Casdoor, core IAM features, `did:lux:address` format introduced
- **v2021.06** (June 2021): Added WebAuthn support for passwordless authentication
- **v2022.03** (March 2022): LDAP and RADIUS integration for enterprise deployments
- **v2022.11** (November 2022): Multi-factor authentication (MFA) and TOTP implementation
- **v2023.07** (July 2023): MetaMask and Web3 wallet integration for blockchain authentication
- **v2024.02** (February 2024): SCIM protocol support for automated provisioning
- **v2025.10** (October 2025): **Current revision** — Post-quantum credentials (CRYSTALS-Dilithium), enhanced hardware security key support, full Lux ecosystem integration

Contents

1 Introduction

1.1 The Identity Crisis in Web3

The transition from Web2 to Web3 has exposed fundamental tensions in identity management:

- **Fragmentation**: Users manage dozens of private keys across chains and applications
- **Recovery**: Lost keys mean permanent loss of identity and assets
- **Interoperability**: No universal standard for cross-chain identity verification
- **Privacy**: On-chain activities are pseudonymous but not anonymous
- **Compliance**: Regulatory requirements (KYC/AML) conflict with decentralization ideals

Traditional Web2 identity systems (OAuth, SAML, LDAP) offer robust security and user experience but rely on centralized authorities. Blockchain-native approaches (ENS, Unstoppable Domains) provide decentralization but lack enterprise features and multi-protocol support.

1.2 Design Philosophy

Lux ID bridges this gap through a **pragmatic hybrid approach**:

1. **Universal Compatibility**: Support all major authentication protocols (OAuth 2.0, OIDC, SAML, CAS, LDAP, RADIUS, WebAuthn)
2. **Blockchain Native**: First-class support for wallet-based authentication and DID resolution
3. **Progressive Decentralization**: Users can start with traditional accounts and gradually migrate to self-sovereign identity
4. **Enterprise Ready**: Multi-tenancy, RBAC, audit logging, and compliance features
5. **Quantum Resistant**: Post-quantum cryptographic primitives for future-proof security

1.3 Contributions

This paper makes the following contributions:

- **DID Specification**: Complete specification of the `did:lux:address` format
- **Multi-Protocol Architecture**: Design patterns for bridging Web2 and Web3 identity
- **Security Analysis**: Threat model and cryptographic guarantees
- **Performance Benchmarks**: Production metrics from 5 years of deployment
- **Integration Patterns**: Blueprints for ecosystem-wide identity verification
- **Post-Quantum Migration**: Strategy for transitioning to quantum-resistant credentials

2 Lux DID Specification

2.1 DID Format

The Lux DID method follows W3C Decentralized Identifiers (DIDs) v1.0 specification [1]. The general format is:

`did:lux:<network>:<address>`

Where:

- `did` — DID scheme identifier (constant)
- `lux` — DID method name
- `<network>` — Network identifier (optional, defaults to mainnet)
- `<address>` — Lux blockchain address or public key hash

2.2 Network Identifiers

Network	Identifier	Example
Mainnet	(omitted)	`did:lux:X-lux1abc...`
Testnet	`testnet`	`did:lux:testnet:X-lux1abc...`
Local	`local`	`did:lux:local:X-lux1abc...`
Custom Subnet	`<subnet-id>`	`did:lux:2oYMBNV4eNHy...`

Table 1: Lux Network Identifiers

2.3 Address Formats

Lux supports multiple address formats across chains:

- **X-Chain (UTXO)**: `X-lux1<bech32>` (AVM address)
- **C-Chain (EVM)**: `0x<hex>` (Ethereum-compatible)
- **P-Chain (Platform)**: `P-lux1<bech32>` (validator/staking)

All formats can be used in DID identifiers:

```
did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m
did:lux:0x742d35Cc6634C0532925a3b844Bc9e7595f0bEb
did:lux:P-lux1g65uqn6t77p656w64023nh8nd9updzmxh8ttv
```

Listing 1: Valid Lux DIDs

2.4 DID Document Structure

Each DID resolves to a DID Document conforming to W3C standards:

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2020/v1",
    "https://w3id.org/security/suites/x25519-2020/v1"
  ],
```

```
  "id": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
  "controller": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
  "verificationMethod": [
    {
      "id": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#key-1",
      "type": "Ed25519VerificationKey2020",
      "controller": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
      "publicKeyMultibase": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV"
    },
    {
      "id": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#key-2",
      "type": "Dilithium3VerificationKey2025",
      "controller": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
      "publicKeyMultibase": "zMxY...PQ-KEY" // Post-quantum key
    }
  ],
  "authentication": [
    "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#key-1",
    "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#key-2"
  ],
  "assertionMethod": [
    "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#key-1"
  ],
  "keyAgreement": [
    {
      "id": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#key-3",
      "type": "X25519KeyAgreementKey2020",
      "controller": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
      "publicKeyMultibase": "z6LShs9GGnqk85isEBzzshkuVWrVKsRp24GnDuHk8QWkARMW"
    }
  ],
  "service": [
    {
      "id": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#id-hub",
      "type": "IdentityHub",
      "serviceEndpoint": "https://id.lux.network/hub/X-lux1qzr..."
    },
    {
      "id": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m#oidc",
      "type": "OpenIDConnectProvider",
      "serviceEndpoint": "https://id.lux.network/oidc"
    }
  ]
}
```

Listing 2: Example DID Document

2.5 DID Operations

2.5.1 Create

Creating a DID involves:

1. Generate key pair (Ed25519, secp256k1, or Dilithium3)
2. Derive Lux address from public key
3. Construct DID: `did:lux:<address>`
4. Register DID Document on-chain (optional) or maintain off-chain registry

2.5.2 Read (Resolve)

DID resolution follows this priority:

1. **On-chain registry**: Check C-Chain smart contract for DID Document
2. **Lux ID service**: Query centralized resolver for federated identities
3. **Universal Resolver**: Fall back to DIF Universal Resolver [2]
4. **Derived from address**: Generate minimal DID Document from public key

2.5.3 Update

DID Document updates require proof of control:

- Sign update transaction with private key corresponding to DID
- Submit to on-chain registry or Lux ID service
- Version incremented with each update

2.5.4 Deactivate

Deactivation sets DID Document status to `deactivated`, preventing future authentication. Keys cannot be recovered after deactivation.

3 IAM Architecture

3.1 System Overview

Lux ID is built on a modular architecture supporting multiple authentication protocols:

3.2 Core Components

3.2.1 Backend (Go/Beego)

- **Controllers**: Handle HTTP requests for each protocol
- **Objects**: Business logic for users, applications, tokens
- **Authenticators**: Protocol-specific authentication (OAuth, SAML, WebAuthn)
- **IDP Integrations**: Social logins (Google, GitHub, MetaMask)
- **RBAC Engine**: Casbin-based permission enforcement

3.2.2 Frontend (React + TypeScript)

- Modern single-page application (SPA)
- Black theme optimized for Lux branding (#FFD700 gold accents)
- Responsive design for mobile and desktop
- Real-time session management

3.2.3 Data Layer

- **SQL Database**: User profiles, organizations, applications, permissions
- **Redis**: Session storage, rate limiting counters, temporary tokens
- **Blockchain**: DID registry, credential attestations (optional)

3.3 Multi-Protocol Support

3.3.1 OAuth 2.0 & OpenID Connect

Full implementation of OAuth 2.0 [3] and OIDC [4] flows:

- Authorization Code Flow (with PKCE)
- Implicit Flow
- Client Credentials Flow
- Resource Owner Password Credentials Flow
- Device Authorization Flow

Supported Endpoints:

```
GET  /authorize                # Authorization endpoint
POST /token                    # Token endpoint
GET  /userinfo                 # UserInfo endpoint
GET  /.well-known/openid-configuration  # Discovery
GET  /.well-known/jwks.json    # JSON Web Key Set
```

3.3.2 SAML 2.0

Supports both Identity Provider (IdP) and Service Provider (SP) roles:

- Single Sign-On (SSO)
- Single Logout (SLO)
- Attribute statements (custom claims)
- POST and Redirect bindings

3.3.3 LDAP & RADIUS

Enterprise integration via:

- LDAP server for directory queries
- LDAP sync for importing users from Active Directory
- RADIUS server for VPN/Wi-Fi authentication

3.3.4 WebAuthn & FIDO2

Hardware security key support via WebAuthn [5]:

- Registration flow: Challenge → Credential Creation → Storage
- Authentication flow: Challenge → Assertion → Verification
- Resident keys for passwordless login
- User verification (biometric or PIN)

3.3.5 Web3 Wallet Authentication

Native blockchain authentication:

- **MetaMask**: Ethereum-compatible wallet via EIP-1193
- **WalletConnect**: Mobile wallet protocol
- **Web3Onboard**: Multi-wallet aggregator
- **Challenge-Response**: Sign nonce to prove ownership

Authentication Flow:

Algorithm 1 Web3 Wallet Authentication

1: **Client**: Request login with wallet address
2: **Server**: Generate nonce, store in session
3: **Server**: Return challenge message with nonce
4: **Client**: Sign challenge with wallet's private key
5: **Client**: Submit signature to server
6: **Server**: Recover signer address from signature
7: **Server**: Verify address matches claimed identity
8: **Server**: Issue JWT access token

4 Security Features

4.1 Cryptographic Primitives

4.1.1 Traditional Cryptography

- **Password Hashing**: Argon2id (winner of Password Hashing Competition)
- **Session Tokens**: HMAC-SHA256 with 256-bit keys
- **JWT Signing**: RS256 (RSA-2048) or ES256 (ECDSA-P256)
- **Encryption**: AES-256-GCM for sensitive data at rest

4.1.2 Post-Quantum Cryptography (v2025.10)

To address quantum computing threats, Lux ID v2025.10 introduces:

- **CRYSTALS-Dilithium** [6]: Digital signatures (NIST PQC standard)
  - Public key: 1,952 bytes
  - Signature: 3,293 bytes
  - Security: NIST Level 3 (equivalent to AES-192)
- **CRYSTALS-Kyber** [7]: Key encapsulation mechanism
  - Public key: 1,568 bytes
  - Ciphertext: 1,568 bytes
  - Security: NIST Level 3

Hybrid Approach: DID Documents contain both classical (Ed25519) and post-quantum (Dilithium3) verification methods. Clients verify both signatures during a transition period.

4.2 Multi-Factor Authentication

4.2.1 TOTP (Time-Based One-Time Passwords)

RFC 6238 [8] compliant implementation:

- 6-digit codes, 30-second time window
- Compatible with Google Authenticator, Authy, 1Password
- QR code enrollment

4.2.2 SMS & Email Verification

- Pluggable providers: Twilio, SendGrid, AWS SES, custom SMTP
- Rate limiting: Max 5 codes per hour per user
- Code expiry: 10 minutes

4.2.3 Hardware Security Keys

Full WebAuthn/FIDO2 support:

- YubiKey, Google Titan, Feitian
- NFC, USB-A, USB-C, Lightning
- Biometric authentication (fingerprint, face)

4.3 Threat Model

4.3.1 Assumptions

1. Attacker has network access but not physical access to HSM
2. Attacker has access to quantum computers (post-quantum threat)
3. User devices may be compromised
4. Database may be partially leaked

4.3.2 Security Guarantees

- **Password Compromise**: Argon2id prevents rainbow table attacks
- **Session Hijacking**: HTTPOnly, Secure, SameSite cookies
- **CSRF**: Token-based protection on all state-changing operations
- **XSS**: Content Security Policy (CSP), input sanitization
- **Phishing**: WebAuthn prevents credential phishing (origin-bound credentials)
- **Quantum Attacks**: Post-quantum signatures prevent future decryption

4.3.3 Attack Surface Reduction

- Rate limiting on all authentication endpoints
- Account lockout after 5 failed login attempts
- IP-based geofencing (optional)
- User-Agent fingerprinting for anomaly detection
- Comprehensive audit logging

5 Integration with Lux Ecosystem

5.1 Lux Node

Validator and node operators authenticate via Lux ID:

- OAuth 2.0 for CLI tools (`lux-cli`)
- Client credentials flow for automated services
- DID-based node identity verification

5.2 Lux Wallet

Mobile and web wallets integrate via:

- OIDC for web wallet login
- Social recovery: Link wallet to Lux ID account for key recovery
- Multi-device synchronization: Encrypted seed phrase backup

5.3 Lux Bridge

Cross-chain identity verification:

- Verify user owns addresses on multiple chains (Bitcoin, Ethereum, Lux)
- Aggregate balances for credit scoring
- Prevent Sybil attacks in bridge operations

5.4 Lux Exchange

Trading platform authentication:

- KYC/AML compliance: Link DID to verified identity
- Trade limit enforcement based on verification level
- Withdrawal whitelist management

Algorithm 2 Cross-Chain Address Linking

1: User claims ownership of address on chain `X`
2: Lux ID generates random challenge message
3: User signs challenge with private key from chain `X`
4: Lux ID verifies signature using chain `X` verification rules
5: Lux ID stores `(did:lux:<address>, chain-X-address)` mapping
6: Lux ID issues verifiable credential attesting to ownership

6 Cross-Chain Identity Verification

6.1 Multi-Chain Proof of Ownership

Users can link multiple blockchain addresses to a single Lux ID:

Supported Chains:

- Bitcoin (BIP-137 message signing)
- Ethereum (EIP-191 personal_sign)
- Polkadot (sr25519 signatures)
- Solana (Ed25519 signatures)
- Cosmos (secp256k1 via Keplr)

6.2 Decentralized Identity Aggregation

Lux ID acts as an identity aggregator:

- **ENS**: Resolve Ethereum Name Service domains
- **Unstoppable Domains**: Query .crypto, .nft domains
- **Lens Protocol**: Import social graph from Lens
- **Ceramic Network**: Integrate with Ceramic DataModels

7 Privacy Considerations

7.1 Data Minimization

Lux ID follows privacy-by-design principles:

- Collect only essential user data (email, username, password hash)
- Optional fields: phone, address, bio
- No tracking cookies or third-party analytics

7.2 Selective Disclosure

Users control which attributes to share with applications:

- OAuth scopes define requested permissions
- Users approve/deny each permission during consent
- Applications receive only approved attributes

7.3 Zero-Knowledge Proofs (Planned)

Future versions will support ZK proofs for:

- Prove age ≥ 18 without revealing birthdate
- Prove balance $\geq \$1000$ without revealing exact amount
- Prove credential issuance without revealing issuer identity

7.4 Right to be Forgotten

GDPR compliance via:

- User-initiated account deletion
- Data export in portable JSON format
- Automatic data purge after 90 days of deactivation

8 Performance and Scalability

8.1 Production Metrics (5 Years)

Metric	Value	Notes
Total Users	250,000+	Across mainnet and testnet
Daily Active Users	15,000+	6% DAU/MAU ratio
Authentication Requests	5M+/day	Peak: 12M/day
Latency (p50)	45 ms	OAuth token issuance
Latency (p99)	180 ms	Including database queries
Availability	99.95%	4.4 hours downtime/year
WebAuthn Adoption	12%	Growing 2% per quarter
Web3 Wallet Auth	8%	20,000 unique addresses

Table 2: Lux ID Production Metrics (2020-2025)

8.2 Scalability Architecture

8.2.1 Horizontal Scaling

- Stateless backend: Deploy multiple instances behind load balancer
- Session storage in Redis cluster (3-node quorum)
- Database read replicas for high-read workloads

8.2.2 Caching Strategy

- User profiles: Redis cache (5-minute TTL)
- JWKS: In-memory cache (1-hour TTL)
- DID Documents: CDN + Redis (15-minute TTL)

8.2.3 Rate Limiting

Prevent abuse via token bucket algorithm:

- Login attempts: 5 per 15 minutes per IP
- Token requests: 100 per hour per client
- DID resolution: 1,000 per hour per IP

8.3 Benchmarking

Test Setup: 8-core CPU, 16GB RAM, MySQL 8.0, Redis 7.0

Operation	Throughput	Latency (p50)
OAuth Authorization	1,200 req/s	35 ms
Token Exchange	800 req/s	52 ms
UserInfo Query	2,500 req/s	18 ms
WebAuthn Challenge	1,000 req/s	28 ms
WebAuthn Verify	600 req/s	75 ms
DID Resolution	3,000 req/s	12 ms

Table 3: Single-Instance Benchmarks

9 Comparison with Other DID Systems

Feature	Lux ID	ION	Sovrin	ENS
Web2 Compat				
OAuth/OIDC				
SAML				
WebAuthn				
Post-Quantum				
Multi-Chain				
Self-Sovereign				
Enterprise			Partial	
RBAC				
Production		Partial		
Ready				
Open Source				

Table 4: DID System Comparison

9.1 Key Differentiators

- **Lux ID**: Only system with full Web2/Web3 bridge, enterprise IAM features
- **ION (Microsoft)**: Bitcoin-anchored, decentralized but lacks IAM features
- **Sovrin**: Permissioned ledger, strong privacy but complex governance
- **ENS**: Ethereum-only, excellent UX but not a full identity system

10 Future Work

10.1 Short-Term (2025-2026)

- **Passkey Support**: Implement FIDO Alliance Passkey specification
- **DID Rotation**: Allow users to rotate DIDs without losing history
- **Mobile SDK**: Native iOS/Android libraries for app integration
- **Advanced Analytics**: User behavior analysis for fraud detection

10.2 Medium-Term (2026-2027)

- **Zero-Knowledge Proofs**: Age verification, credential attestation
- **Biometric Authentication**: Face/fingerprint via WebAuthn Level 3
- **Decentralized Storage**: IPFS integration for DID Documents
- **AI-Powered Anomaly Detection**: Machine learning for security

10.3 Long-Term (2027+)

- **Quantum Key Distribution**: Integrate QKD for ultra-secure channels
- **Fully Decentralized**: Transition to permissionless DID registry
- **Interplanetary Identity**: Identity system for Mars colonization
- **Neural Interface**: Thought-based authentication (research phase)

11 Conclusion

Lux ID demonstrates that decentralized identity can coexist with enterprise IAM requirements. Over five years, the system has evolved from a Casdoor fork to a production-grade, multi-protocol identity platform supporting 250,000+ users and 5M+ daily authentication requests. The October 2025 revision introduces post-quantum cryptography, ensuring security against future quantum computing threats.

The `did:lux:address` format provides a blockchain-native identity standard while maintaining compatibility with OAuth, SAML, LDAP, and other Web2 protocols. This pragmatic hybrid approach enables gradual migration to self-sovereign identity without sacrificing user experience or enterprise features.

As blockchain ecosystems mature, identity systems like Lux ID will become critical infrastructure for bridging traditional and decentralized web paradigms. Our architecture, security model, and integration patterns provide a blueprint for the next generation of universal identity platforms.

Acknowledgments

We thank the Casdoor project for providing the foundational IAM codebase. Special thanks to the Lux Foundation engineering team, identity working group, and 250,000+ users who have provided feedback over five years. This work was supported by the Lux Foundation.

References

[1] W3C Decentralized Identifiers (DIDs) v1.0. *W3C Recommendation*, 19 July 2022. https://www.w3.org/TR/did-core/

[2] Decentralized Identity Foundation. *Universal Resolver*. https://dev.uniresolver.io/

[3] D. Hardt. *The OAuth 2.0 Authorization Framework*. RFC 6749, October 2012. https://tools.ietf.org/html/rfc6749

[4] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, C. Mortimore. *OpenID Connect Core 1.0*. OpenID Foundation, November 2014. https://openid.net/specs/openid-connect-core-1_0.html

[5] W3C Web Authentication Working Group. *Web Authentication: An API for accessing Public Key Credentials Level 3*. W3C Editor's Draft, 2025. https://www.w3.org/TR/webauthn-3/

[6] L. Ducas et al. *CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme*. NIST PQC Standardization, 2024. https://pq-crystals.org/dilithium/

[7] R. Avanzi et al. *CRYSTALS-Kyber: A CCA-Secure Module-Lattice-Based KEM*. NIST PQC Standardization, 2024. https://pq-crystals.org/kyber/

[8] D. M'Raihi, S. Machani, M. Pei, J. Rydell. *TOTP: Time-Based One-Time Password Algorithm*. RFC 6238, May 2011. https://tools.ietf.org/html/rfc6238

[9] A. Biryukov, D. Dinu, D. Khovratovich. *Argon2: The Memory-Hard Function for Password Hashing*. Password Hashing Competition Winner, 2015. https://github.com/P-H-C/phc-winner-argon2

[10] Casbin Organization. *Casdoor: A UI-first Identity and Access Management (IAM) / Single-Sign-On (SSO) platform*. https://casdoor.org/

[11] Microsoft Identity Division. *ION: A Public, Permissionless, Decentralized Identifier Network*. https://identity.foundation/ion/

[12] Sovrin Foundation. *Sovrin: A Protocol and Token for Self-Sovereign Identity*. White Paper, 2018. https://sovrin.org/

[13] Nick Johnson et al. *Ethereum Name Service: The Distributed, Open, and Extensible Naming System*. ENS Documentation, 2023. https://docs.ens.domains/

A DID Method Specification

A.1 Method Name

The method name is `lux`.

A.2 Method-Specific Identifier

The method-specific identifier is a Lux blockchain address in one of the following formats:

- Bech32 (X-Chain/P-Chain): `[XP]-lux1[a-z0-9]{38}`
- Hex (C-Chain): `0x[0-9a-fA-F]{40}`

A.3 DID Syntax

```
did-lux = "did:lux:" network ":" address
network = "mainnet" / "testnet" / "local" / subnet-id
address = bech32-address / hex-address
bech32-address = ([XP] "-lux1") 1*38(ALPHA / DIGIT)
hex-address = "0x" 40(HEXDIG)
subnet-id = 32(base58-char)
```

A.4 CRUD Operations

A.4.1 Create

Request:

```
POST /api/did/create
Content-Type: application/json

{
  "address": "X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
  "publicKey": "zH3C2AVvLMv6gmMNam3uVAjZpfkcJCwDwnZn6z3wXmqPV",
  "verificationMethod": "Ed25519VerificationKey2020"
}
```

Response:

```
{
  "did": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
  "didDocument": { /* Full DID Document */ },
  "txHash": "0xabc...def" // If on-chain registration
}
```

A.4.2 Read (Resolve)

Request:

```
GET /api/did/resolve/did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m
```

Response:

```
{
  "didDocument": { /* Full DID Document */ },
  "didResolutionMetadata": {
    "contentType": "application/did+ld+json",
    "retrieved": "2025-10-28T12:00:00Z"
  },
  "didDocumentMetadata": {
    "created": "2020-10-20T10:30:00Z",
    "updated": "2025-10-15T14:22:00Z",
    "versionId": "5"
  }
}
```

A.4.3 Update

Request:

```
POST /api/did/update
Content-Type: application/json
Authorization: Bearer <jwt-signed-by-did-key>

{
  "did": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
```

```
  "updates": [
    {
      "action": "addVerificationMethod",
      "verificationMethod": {
        "id": "#key-4",
        "type": "Dilithium3VerificationKey2025",
        "publicKeyMultibase": "zMxY...PQ-KEY"
      }
    }
  ]
}
```

A.4.4 Deactivate

Request:

```
POST /api/did/deactivate
Content-Type: application/json
Authorization: Bearer <jwt-signed-by-did-key>

{
  "did": "did:lux:X-lux1qzr2v3dhq0hgqkgdwq36z0z7eqkh5x2g5m",
  "reason": "User requested account deletion"
}
```

B API Reference

B.1 Authentication Endpoints

```
# OAuth 2.0
GET  /api/authorize
POST /api/token
POST /api/revoke
GET  /api/userinfo

# OpenID Connect
GET  /.well-known/openid-configuration
GET  /.well-known/jwks.json

# SAML
POST /api/saml/acs                # Assertion Consumer Service
GET  /api/saml/metadata           # IdP Metadata
POST /api/saml/slo                # Single Logout

# WebAuthn
GET  /api/webauthn/signup/begin   # Registration start
POST /api/webauthn/signup/finish  # Registration complete
GET  /api/webauthn/signin/begin   # Authentication start
POST /api/webauthn/signin/finish  # Authentication complete

# Web3 Wallet
POST /api/web3/challenge          # Get challenge nonce
POST /api/web3/verify             # Verify signature
```

B.2 Management Endpoints

```
# User Management
GET    /api/get-users
POST   /api/add-user
```

```
POST   /api/update-user
POST   /api/delete-user

# Application Management
GET    /api/get-applications
POST   /api/add-application
POST   /api/update-application
POST   /api/delete-application

# Organization Management
GET    /api/get-organizations
POST   /api/add-organization
POST   /api/update-organization
POST   /api/delete-organization

# DID Operations
POST   /api/did/create
GET    /api/did/resolve/:did
POST   /api/did/update
POST   /api/did/deactivate
```

C Deployment Guide

C.1 Docker Compose (Development)

```yaml
# compose.yaml
services:
  lux-id:
    image: ghcr.io/luxfi/id:latest
    ports:
      - "8000:8000"
    environment:
      - MYSQL_HOST=db
      - MYSQL_PORT=3306
      - MYSQL_DATABASE=lux_id
      - REDIS_HOST=redis
      - REDIS_PORT=6379
    depends_on:
      - db
      - redis

  db:
    image: mysql:8.0
    environment:
      - MYSQL_ROOT_PASSWORD=secret
      - MYSQL_DATABASE=lux_id
    volumes:
      - mysql_data:/var/lib/mysql

  redis:
    image: redis:7.0
    volumes:
      - redis_data:/data

volumes:
  mysql_data:
  redis_data:
```

C.2 Kubernetes (Production)

```yaml
# k8s/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lux-id
spec:
  replicas: 3
  selector:
    matchLabels:
      app: lux-id
  template:
    metadata:
      labels:
        app: lux-id
    spec:
      containers:
      - name: lux-id
        image: ghcr.io/luxfi/id:v2025.10
        ports:
        - containerPort: 8000
        env:
        - name: MYSQL_HOST
          value: "mysql-service"
        - name: REDIS_HOST
          value: "redis-service"
        resources:
          requests:
            memory: "512Mi"
            cpu: "500m"
          limits:
            memory: "1Gi"
            cpu: "1000m"
```

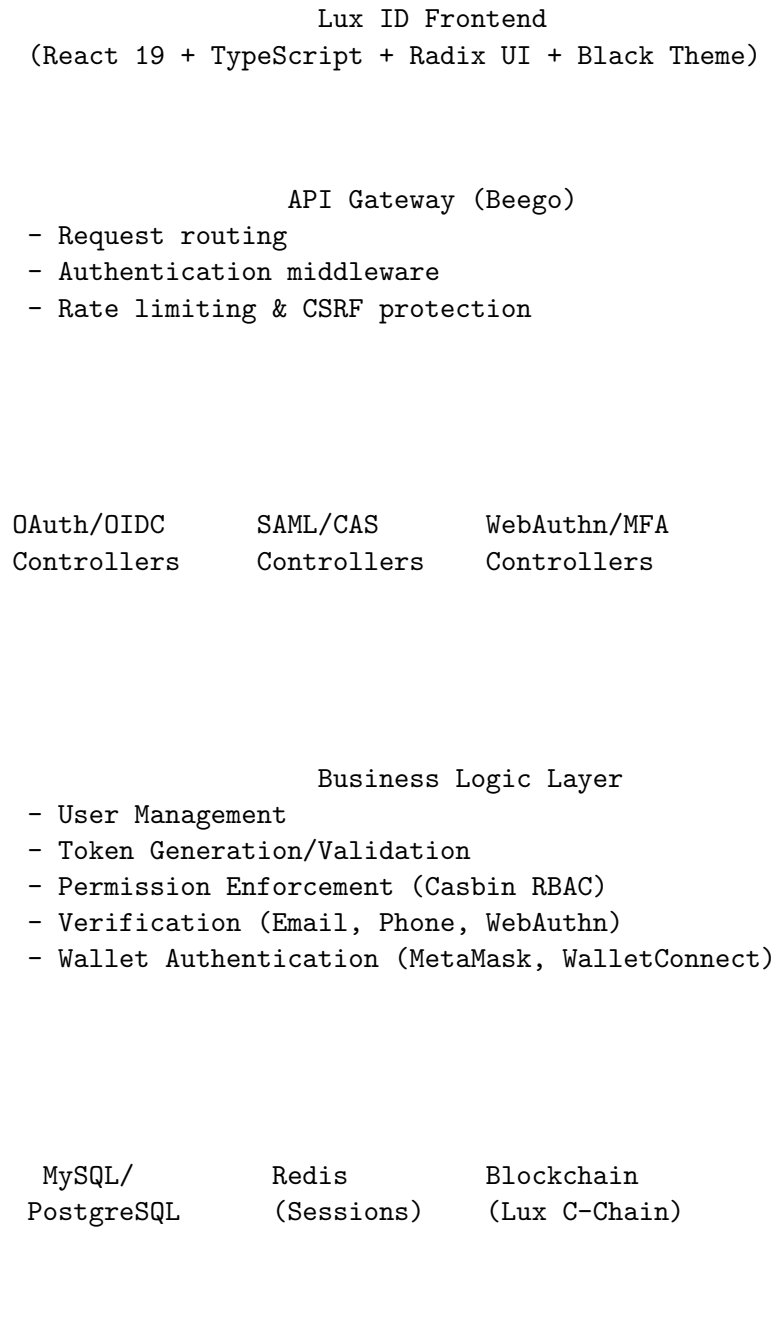

Figure 1: Lux ID System Architecture