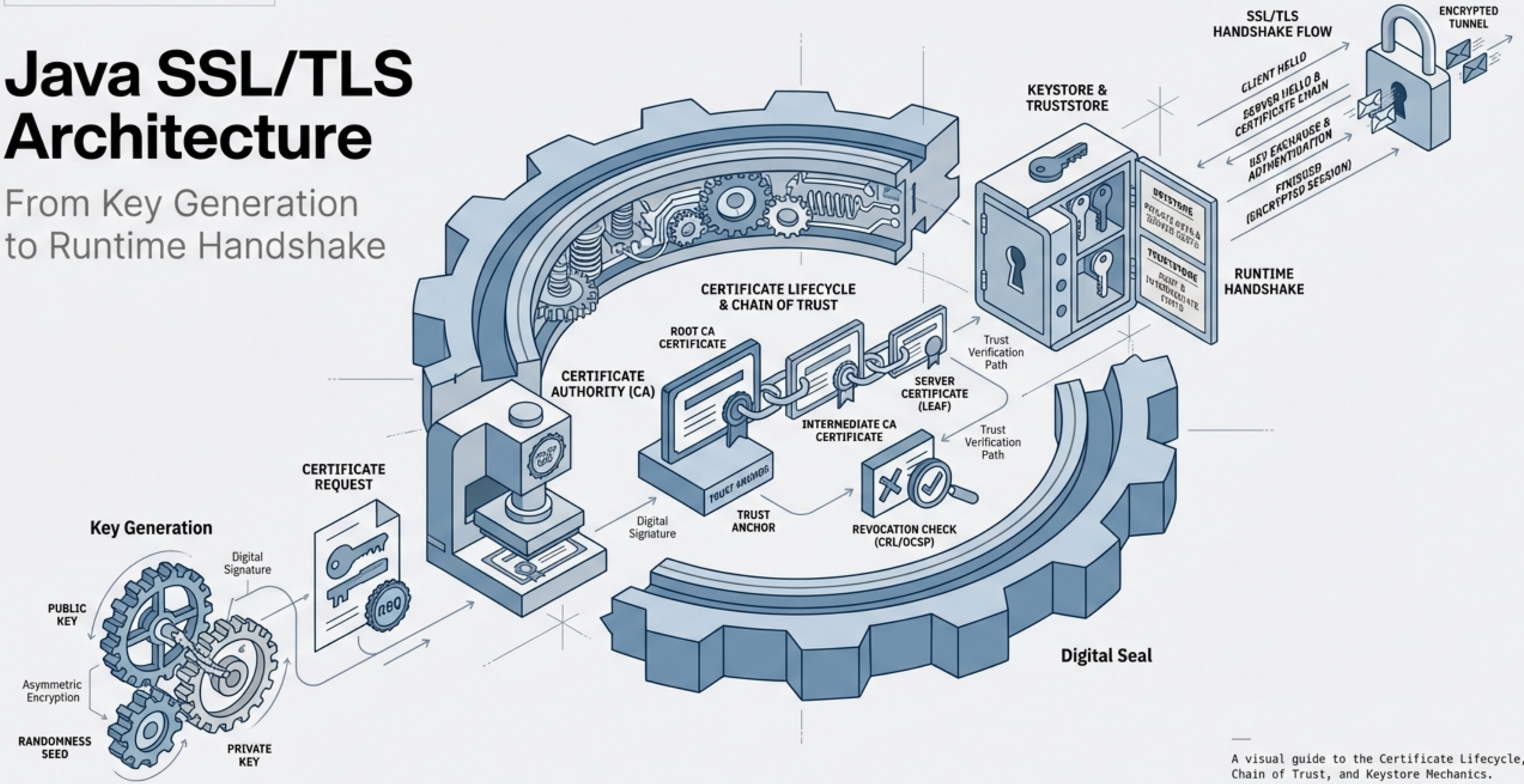
READING DECK // REFERENCE

Java SSL/TLS Architecture

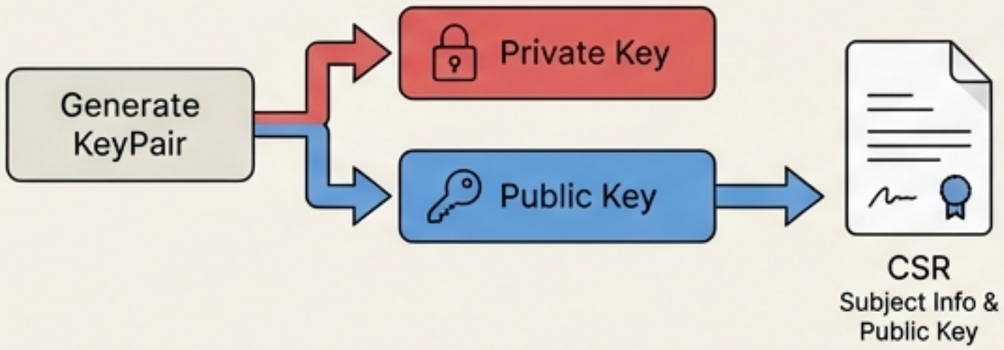
From Key Generation to Runtime Handshake



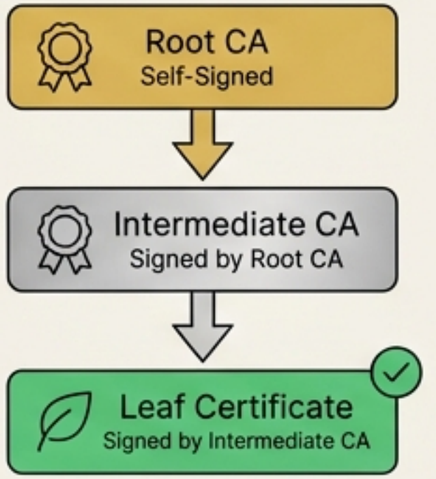
A visual guide to the Certificate Lifecycle, Chain of Trust, and Keystore Mechanics.

The Complete Architecture

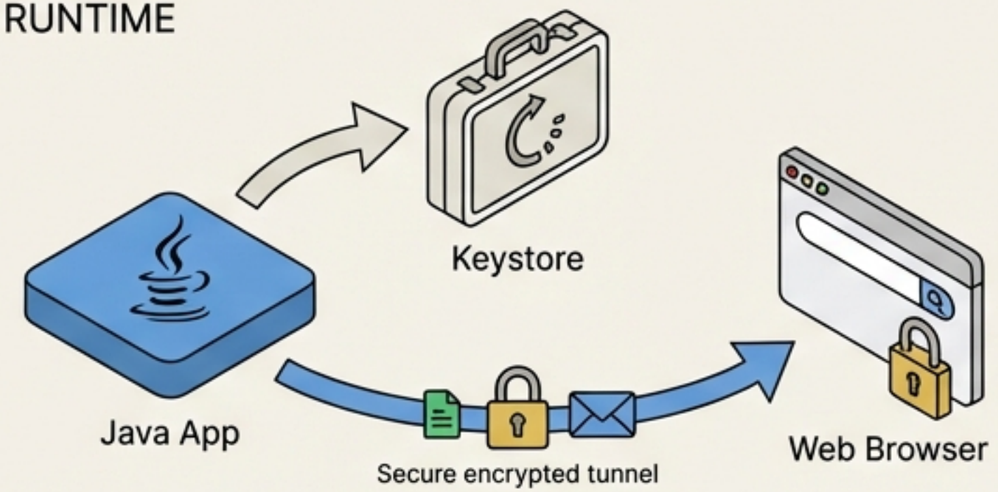
01. GENERATION



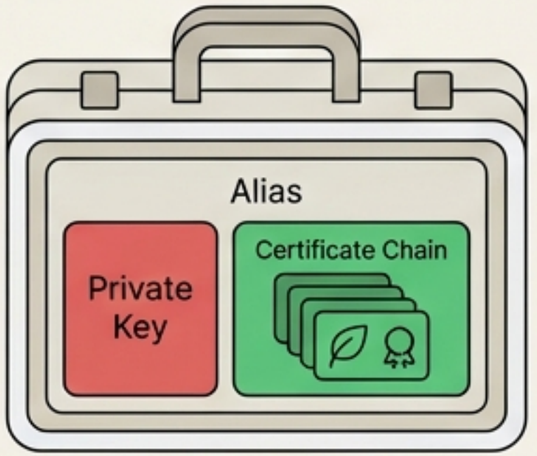
02. TRUST CHAIN



04. RUNTIME



03. STORAGE



We will zoom into each quadrant of this process to demystify how Java establishes a secure connection.

The Identity and The Lock

Generate KeyPair

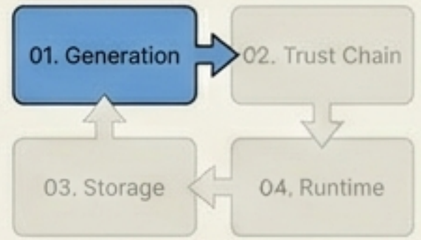
Private Key

Kept secret. If stolen, the server can be impersonated.

Public Key

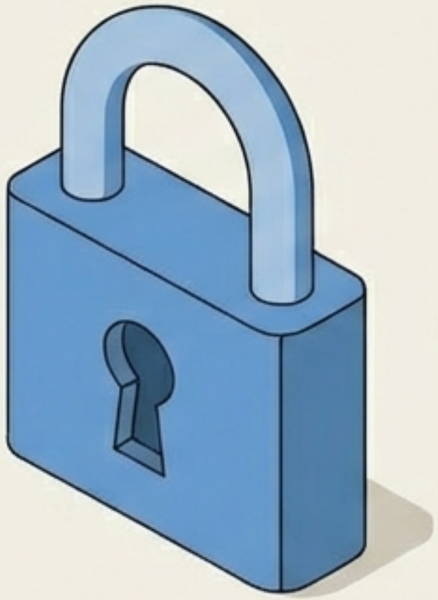
Distributed freely. Used to encrypt messages that only the Private Key can decrypt.

The Certificate Signing Request (CSR)



The bridge between key generation and certification.

Public Key



+

Identity Metadata

CN: Common Name

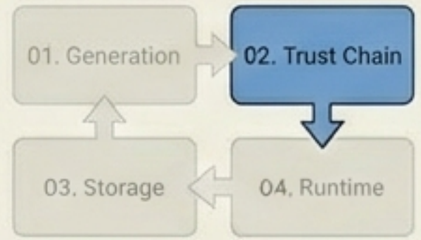
O: Organization

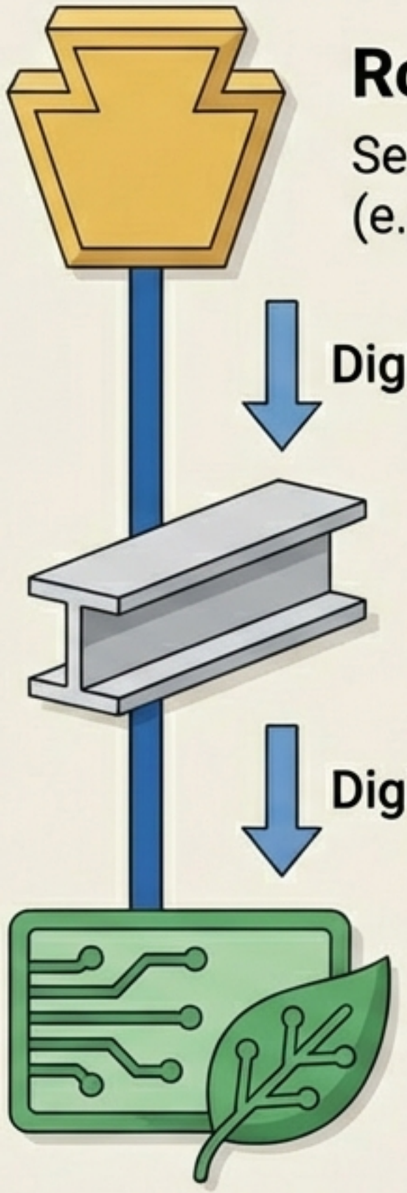
C: Country

=

"You send this file to a Certificate Authority (CA) to say, "Please verify who I am and sign my public key.'"

The Hierarchy of Trust





Root CA

Self-Signed. The Anchor of Trust
(e.g., DigiCert Global Root).

Digital Signature

Intermediate CA

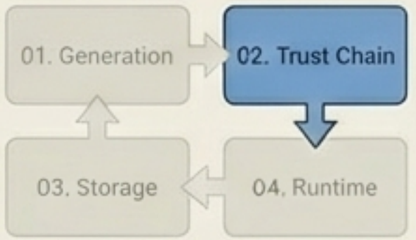
Signed by Root. The Operational Authority.

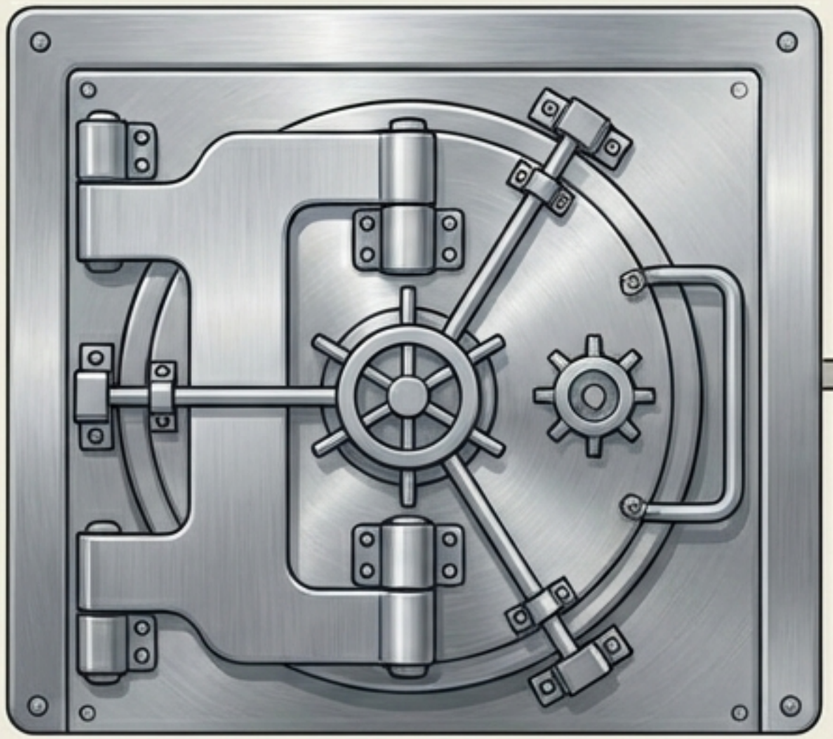
Digital Signature

Leaf / Server Certificate

Signed by Intermediate. Your Identity
(e.g., www.example.com).

Why We Need Intermediate CAs





Delegated Authority

Root CA = The CEO

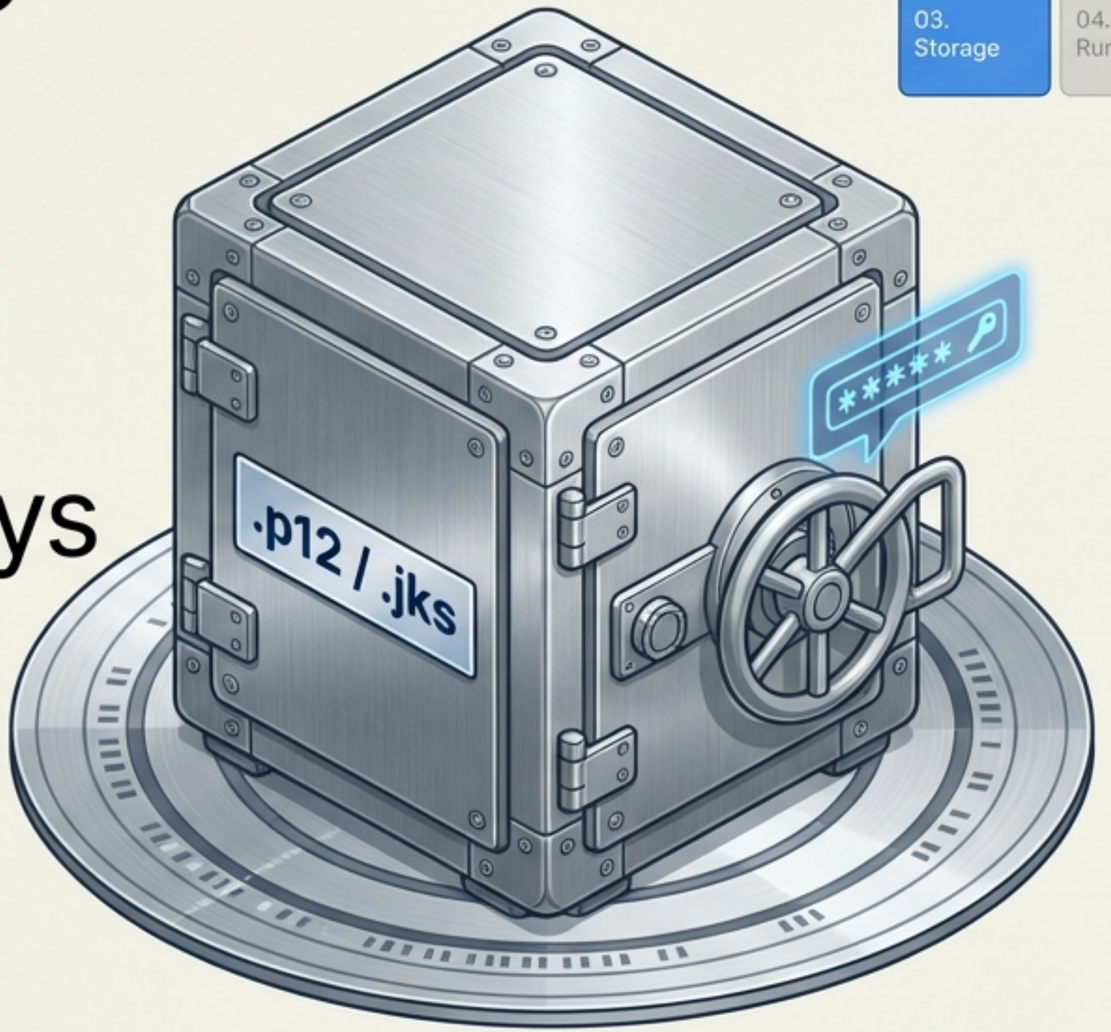
Keys are offline in high-security vaults.
Too risky to use for daily tasks.

Intermediate CA = The Manager

Handles the daily work of verifying and signing.
If compromised, it can be revoked without destroying the Root.

The Java Keystore

A password-protected file format used by Java to store cryptographic keys and certificates.



This file is the resting place for the digital identity before the application starts.

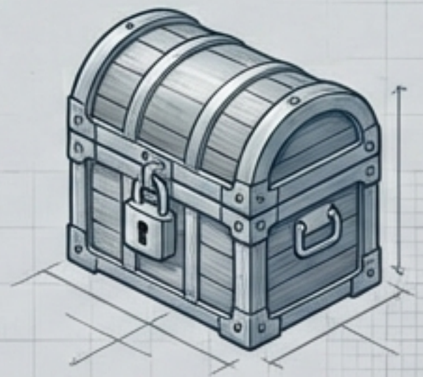
File Formats: JKS vs. PKCS12

JKS (Java KeyStore)

.jks

DEPRECATED / LEGACY

Proprietary format specific to Java. **Avoid** for new projects.



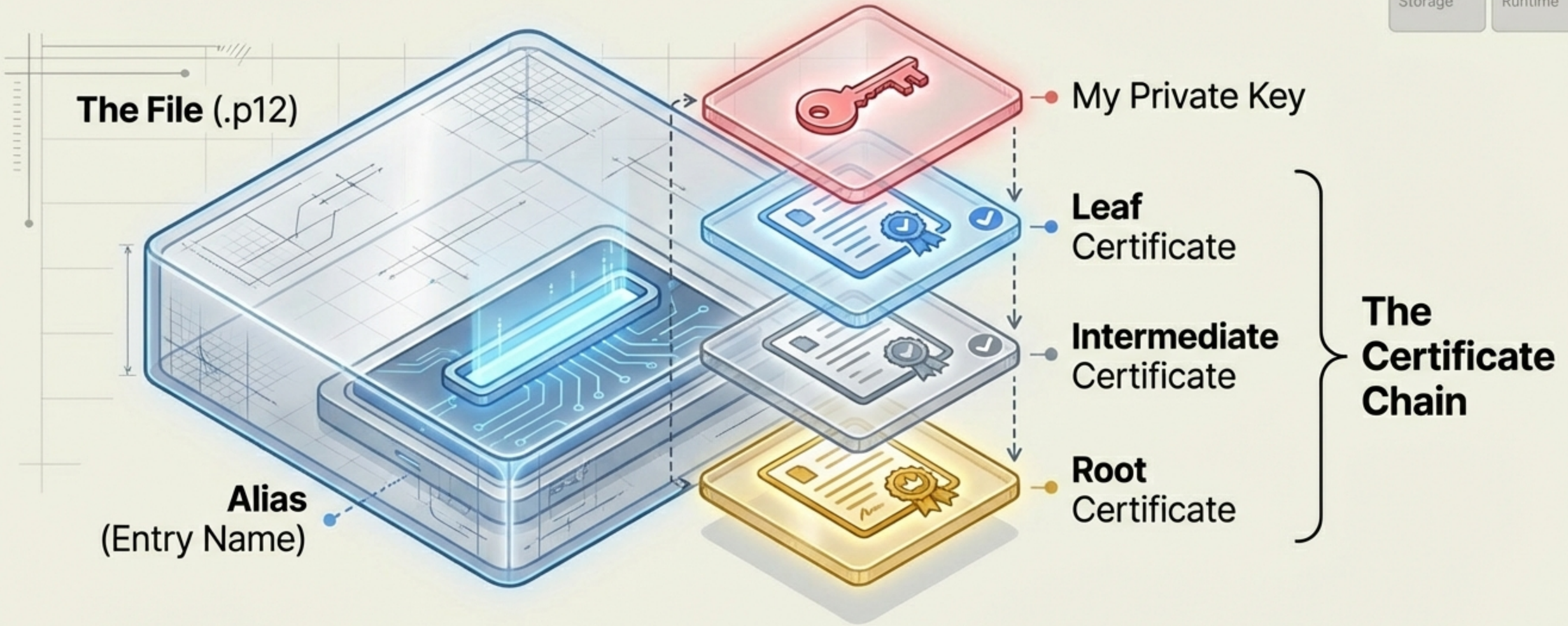
PKCS12

.p12 or **.pfx**

RECOMMENDED / STANDARD

Default since Java 9. Industry-standard format compatible with **OpenSSL**, **Windows**, and browsers.

Anatomy of a Keystore Entry



The Keystore stores the entire chain so the server can prove its identity fully.

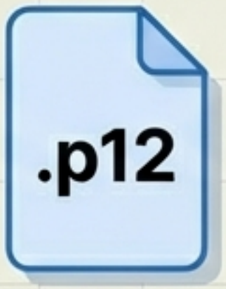
Application Startup

Java App
(Tomcat/Spring)

Java App
(Tomcat/Spring)

Load File

Java KeyStore (JKS) or PKCS12 loaded into JVM



Decrypt

Keystore password is provided (e.g., from config or env var)

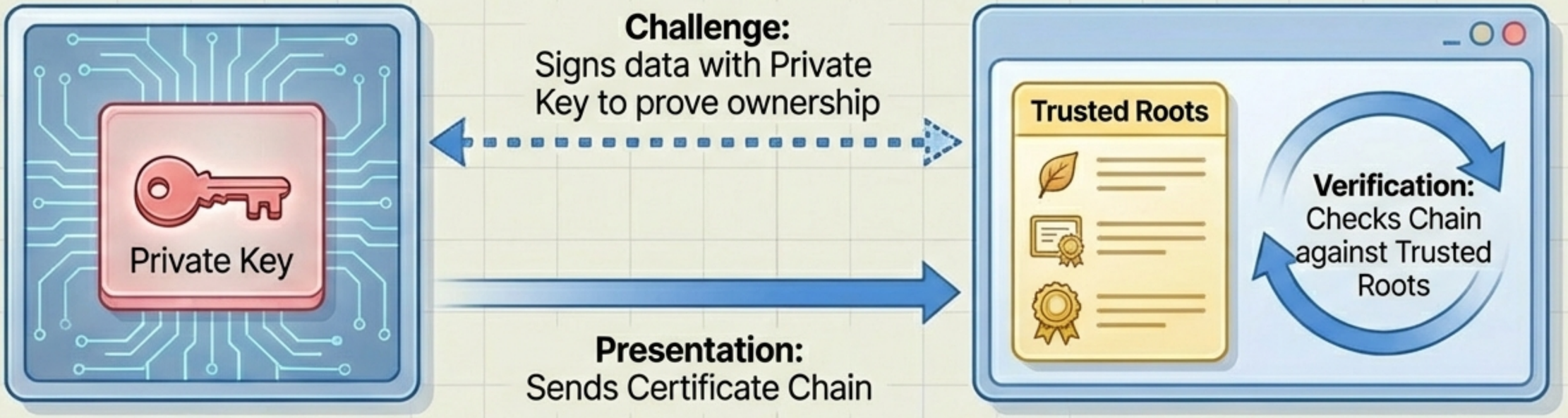
Alias specified in configuration is identified

Extract Alias

Private key and certificate chain are loaded for SSL/TLS communication

Private Key + Chain

The Runtime Handshake



Java Application
(Server)

Web Browser
(Client)

The server presents its credentials, and the browser verifies them against its known authorities.

Crucial Distinction: Keystore vs. Truststore

KEYSTORE

"Who am I?"

↓

My Private Keys + My Public Certs

↓

Serving HTTPS to others

File Name: keystore.p12

TRUSTSTORE

"Who do I trust?"

↓

Others' Public Certs (CA Roots)

↓

Connecting to other HTTPS services

File Name: cacerts

The Architecture Recap

PRO TIP: Always prefer PKCS12 (.p12) over JKS for modern Java applications to ensure compatibility.