Adopting OpenSSL for Enterprise Software

Thanigai Nallathambi

08 Oct 2025

Oracle Corporation

Agenda

Challenges in a large-scale, diverse environment

Oracle's migration path and solutions

Recommendations for future OpenSSL development

Enterprise landscape: Scale and Diversity

Leading global enterprise software and cloud infrastructure vendor

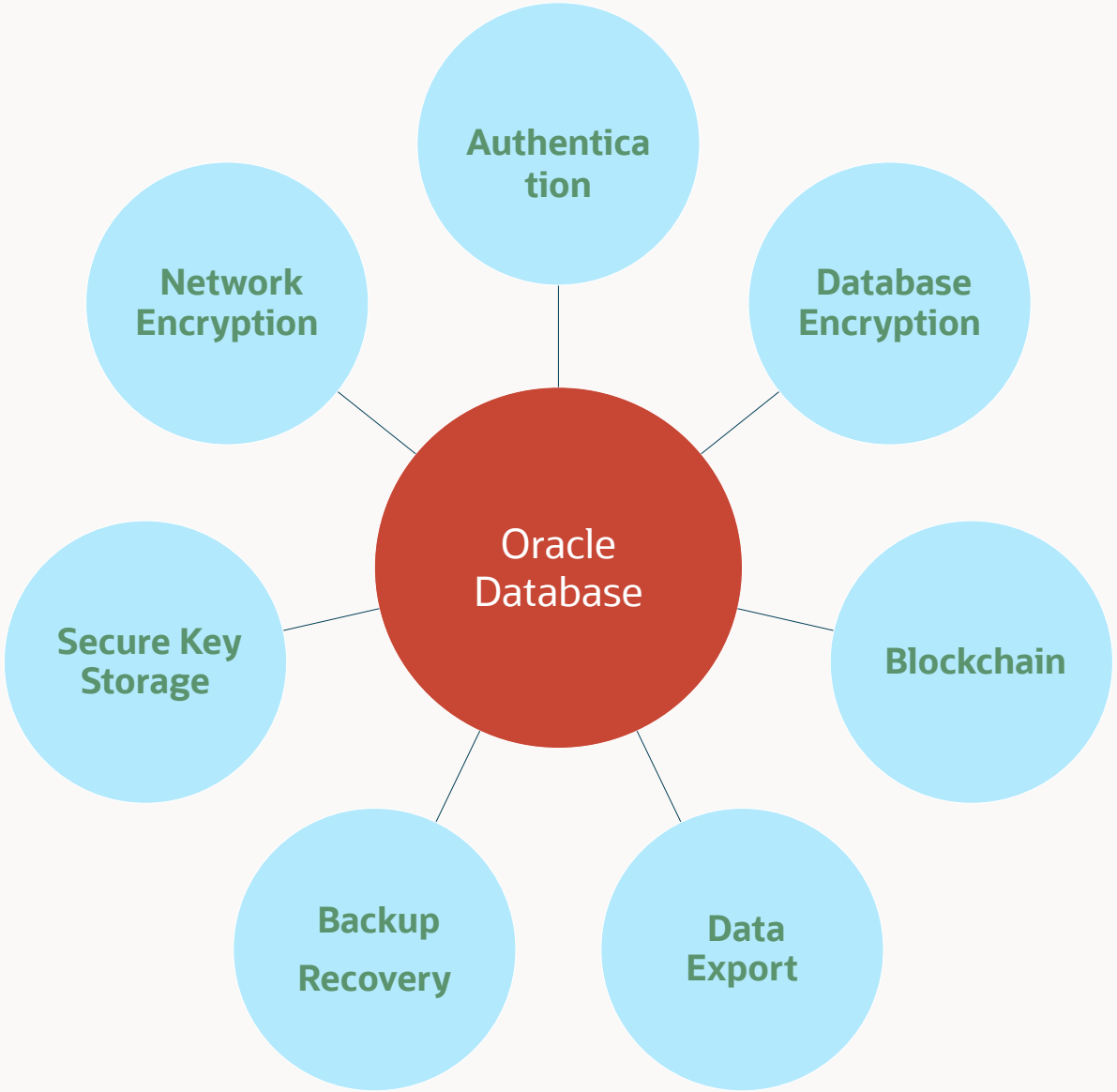
Extensive portfolio: operating systems, databases, applications, cloud

Environments span from small to very large

Enterprise landscape: Core Requirements

- **Security:** Strong cryptography and data integrity
- **Performance:** High scalability and low latency
  - hundreds of thousands of connections with <100 ms connection time
- **Modularity**: Allow easy addition of new algorithms/features
  - Patching and migration requires smooth transition
- **Stability:** Reliable operations and robust error handling under extreme load
- **Compliance**: FIPS certification mandatory
- **Platforms**: Support a variety of platforms (Linux, Windows, AIX, …)
- **Backward compatibility**: New client/server versions should work with older counterparts
  - Older version of OpenSSL 1.0.2, 1.1.1 and older version of TLS 1.1 and TLS 1.2
  - Interoperation with Java crypto stack from JDK 11 to JDK 25
- **Long-term support**: Released software supported for long time, and security fixes must be backportable

Use of cryptography in Oracle Databases



FIPS crypto required for all features

Why use OpenSSL in Oracle database

Supports most existing use cases in Oracle Database

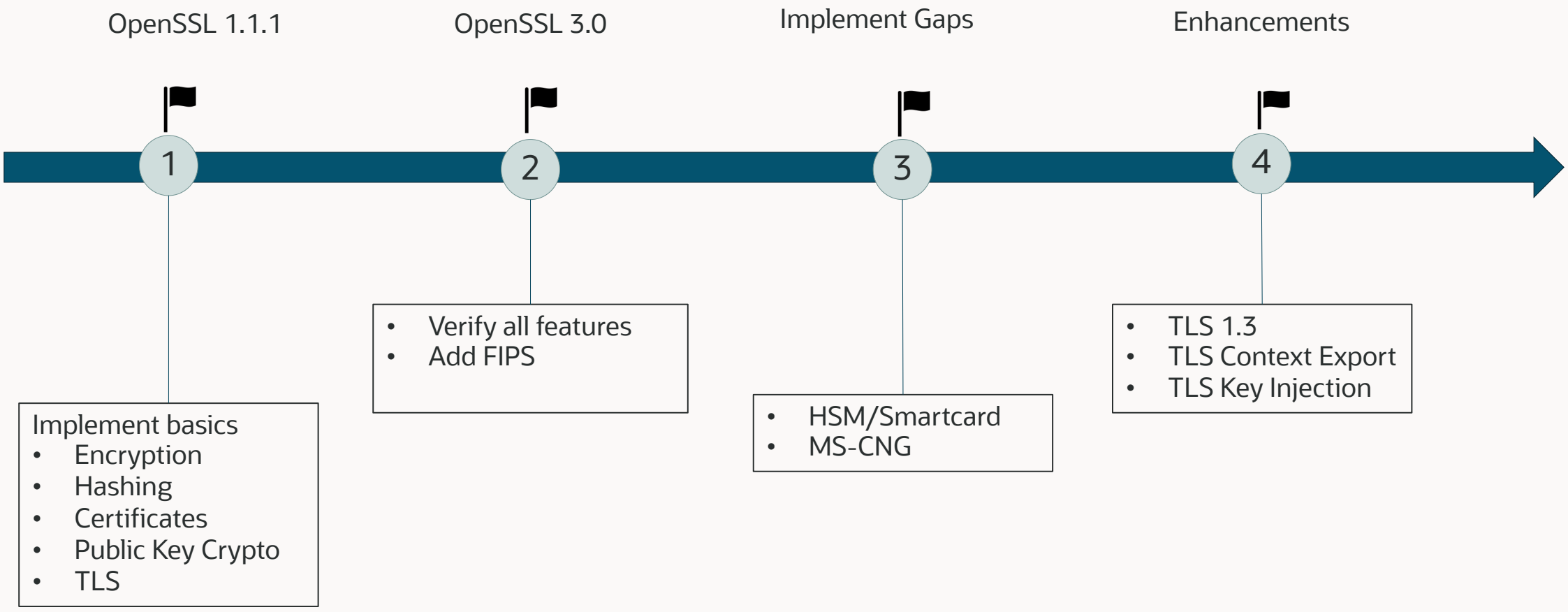
Proven performance in enterprise server applications

Flexible architecture in OpenSSL 3.0 enables future enhancements

Improved performance with native hardware acceleration

Strong industry support and participation

The journey of OpenSSL adoption : Execution



Oracle's efforts to address new feature requirements and gaps

Oracle enhances OpenSSL to address enterprise needs and is ready to contribute to the community!

TLS with private keys on external storage – **Large enterprises are required to use HSMs for key storage**

- Sign using private key
- Implement EXTKS (External Keystore) provider to support Microsoft Certificate store/CNG (Crypto Nextgen); extensible for other OS key stores
- Implement PKCS11 provider to support HSMs and smartcards

Context export for transparent connection handoff – **Fast TLS connections with connection pool is a must**

- Digest context export
- TLS context export

TLS key injection for enhanced security - **Provide quantum resistance for TLS 1.2**

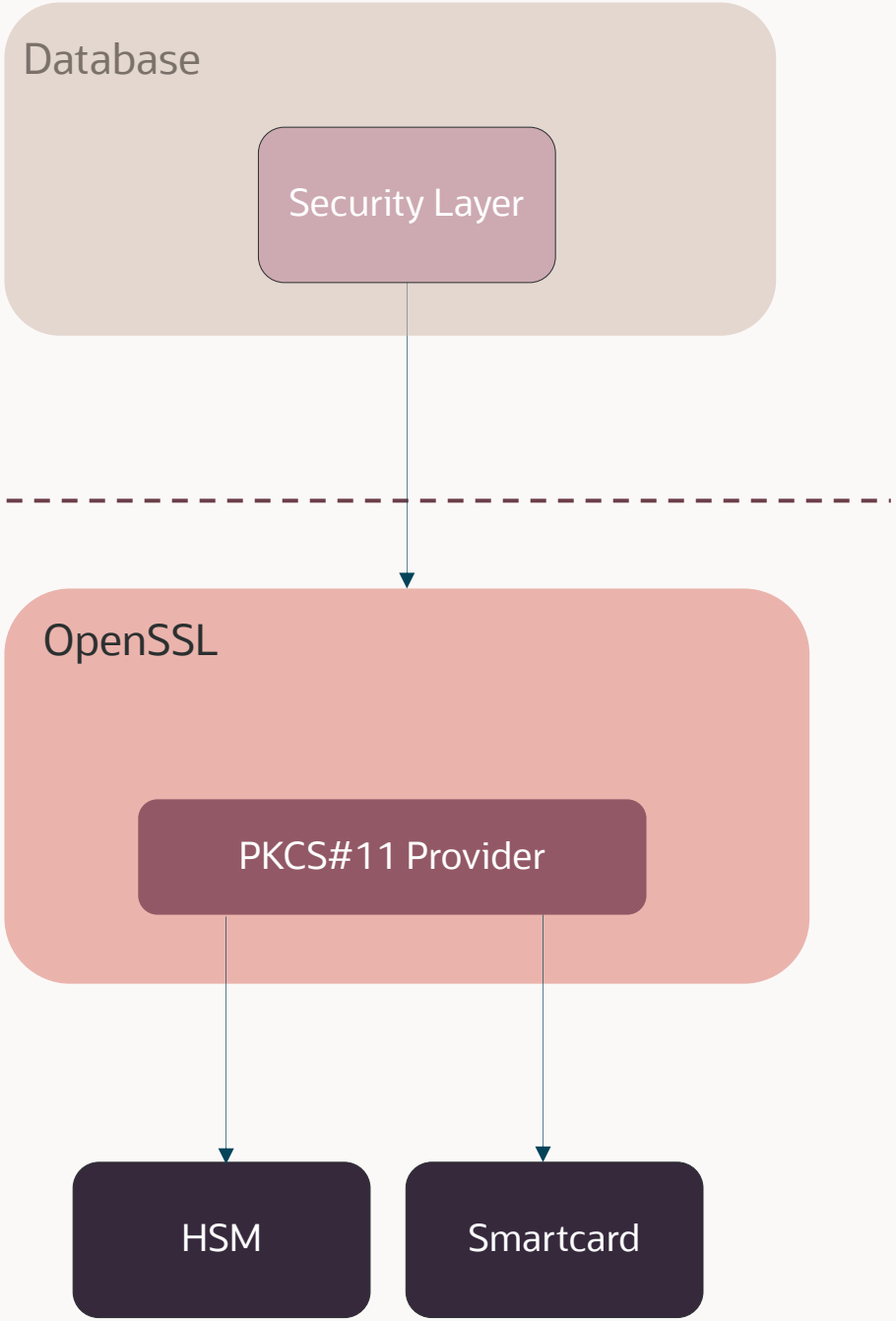
- Strengthen session key with a secret shared between client and server

PKCS#11

Requirement: Support sign/verify operations during TLS using private keys held in **HSMs and smartcards**

Challenge: No native, built-in PKCS#11 provider in OpenSSL

Oracle Solution: Implemented custom OpenSSL provider to access credentials on HSMs and smartcards via PKCS#11



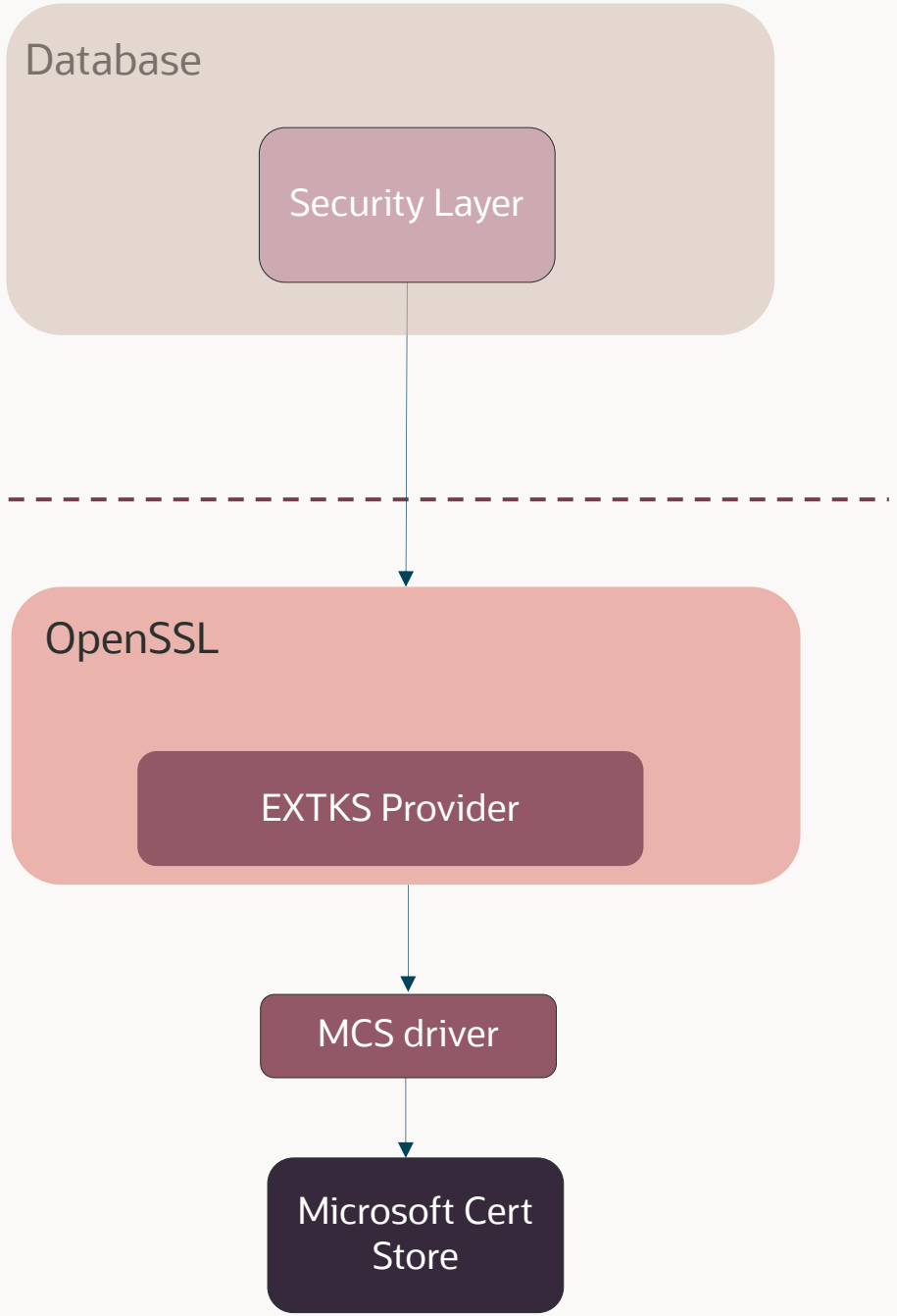
OS Certificate Store

Requirement:
- Support for TLS using certificates on Microsoft Certificate Store (MCS)
- Support Microsoft Crypto Nextgen (CNG) API

Challenge: No native support for MCS in OpenSSL

Oracle solution: Developed the External Key Store (EXTKS) provider to access credentials in MCS

Solution is generic and extensible to support additional OS platforms such as MacOS keychains



TLS context export

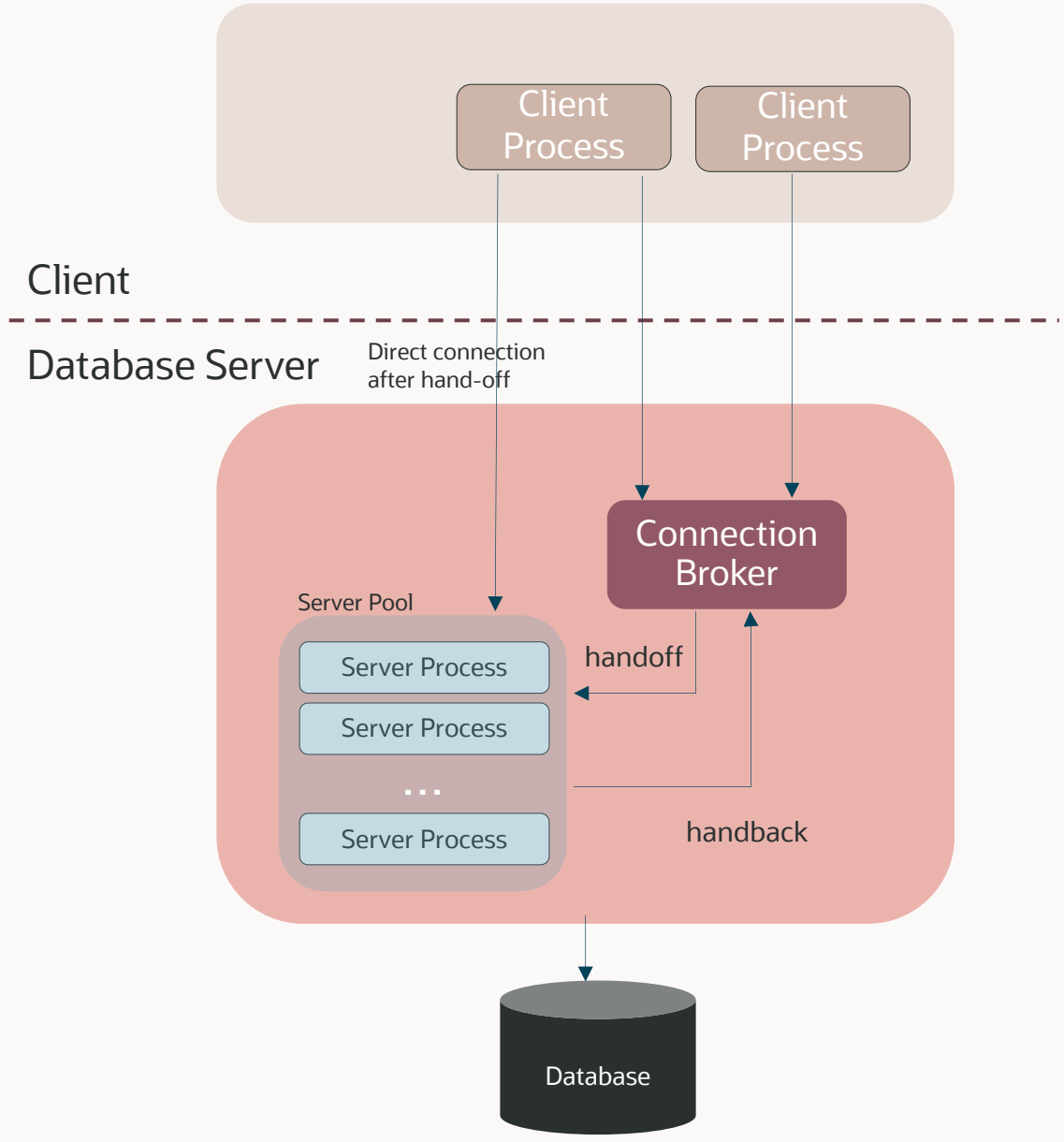
Multiple database clients access the database through **a pool of server processes**

Long running client connections may be switched between different server processes

TLS connections should be resumable on any server process without a new handshake

Requires transferring TLS state along with the socket to remote server process

Seamless and transparent to the client



Application key injection into TLS

Benefits:
- Adding an out of band shared secret prevents attacks on key exchange and provide quantum resistance

Adds an additional layer of security by reinforcing TLS keys with a client/server shared secret

Client and server establish shared secret outside of TLS

TLS session keys are updated using the shared secret without requiring a new handshake
- Added API to accept application secret to local context
- Mix secret with session keys on both sides
- TLS 1.3: Use TLS “KeyUpdate” message to request the other side to update key

Issues and challenges

Documentation
- Some behaviors are not well documented; often require reading source code

Performance
- Initialization overhead: OpenSSL mapping tables and error strings (*pool of servers to reduce latency*)
- AES slower than Intel's IPP, improved in 3.5
- Threads: excessive locking in encryption calls (algorithm fetching), fixed in 3.5
- Memory: timing cleanup is a challenge (*partly due to complexity of the database architecture*)

Debugging
- Complex source code (e.g. ASN.1 macros) is difficult to debug

Issues and challenges

Price of popularity

- Greater exposure to CVEs due to widespread use (*and short time to react!*)
- Security fixes must be implemented rapidly across all deployments
- Require dedicated processes for timely absorption and delivery of security updates

Recommendations: Areas for improvement

Thread-safety and performance under high concurrency

Enhancements to the provider architecture

Better documentation for complex use cases

Extensive testing of enterprise use cases to ensure performance and reliability

Comprehensive tracing and diagnosability


ORACLE