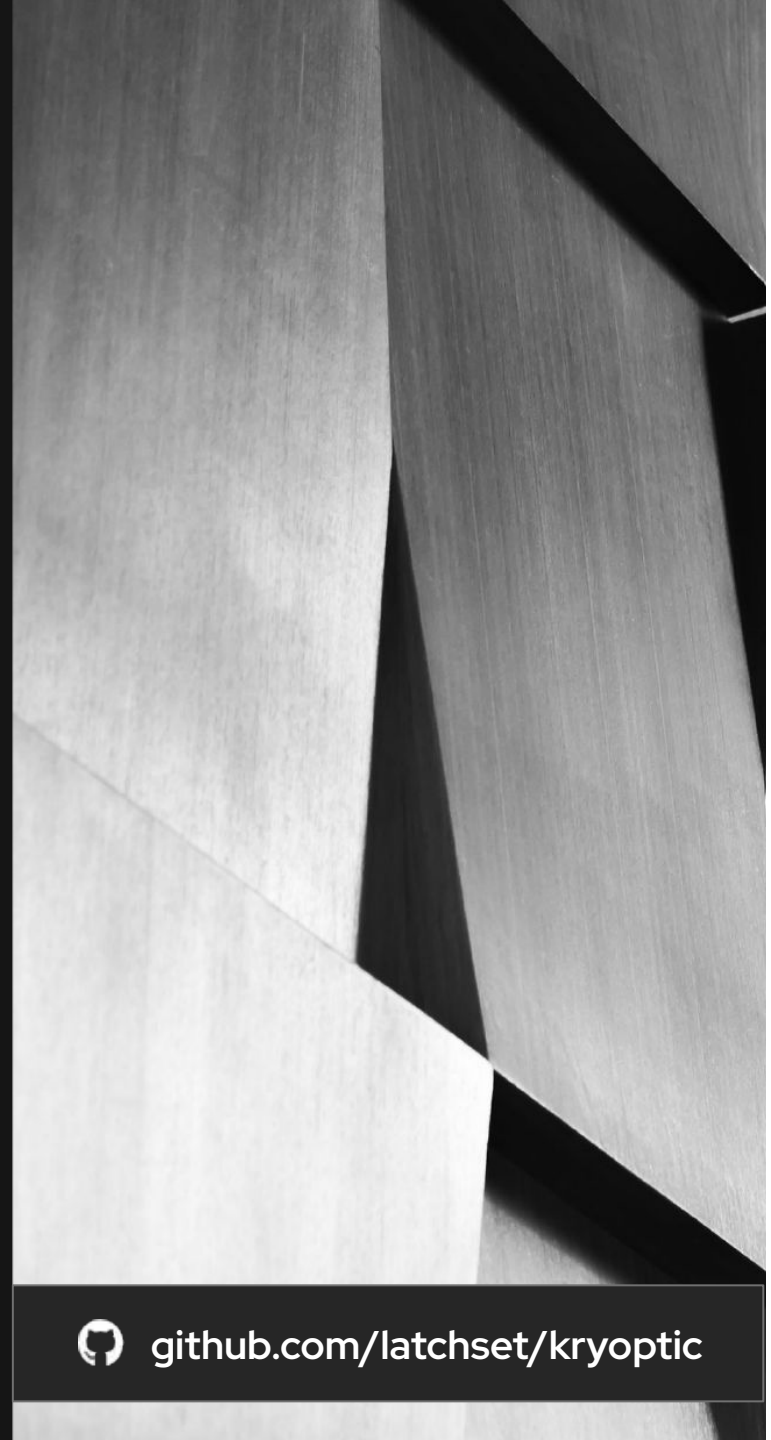


github.com/latchset/kryoptic

Hybrid FIPS Module Using OpenSSL Code

The Kryoptic Project

Simo Sorce

Distinguish Engineer

Red Hat

PROJECT CONTEXT

The Goals of “--features fips”



Self-Contained Module

A single dynamic library replacing complex, multi-libraries cryptographic setups.



Dual-API Capability

Simultaneous exposure of the OpenSSL provider internal API and PKCS#11 driver API.



Unified Core

Avoiding multiple separate modules by sharing one certified cryptographic foundation.

INTERFACE STRATEGY

One Module, Multiple Applications

The FIPS Bottleneck

Certifying independent cryptographic modules for different application ecosystems is highly resource-intensive.

Unified Certification

A single FIPS 140-3 validation covers the underlying core for all consumers.

Broad Ecosystem Reach

OpenSSL and PKCS#11 applications run on the exact same certified codebase.

Operational Simplicity

Administrators deploy one dynamic library for vastly different application stacks.

DEPLOYMENT ADVANTAGE

The Single Binary Model

1

Universal Binary Build

Minimal Footprint

External dependencies reduced to the kernel and libc.

Static Compilation

FIPS cryptographic core embedded directly into the library.

Broad Compatibility

Targets multiple Linux (RHEL) operating system versions simultaneously.

HYBRID STRATEGY

Distinct Entry Paths, Shared Core

Path A: OpenSSL Provider

Thin C wrapper around OSSL_provider_init_int.

Path B: PKCS#11 Driver

Rust implementation handles full PKCS#11 architecture.





Statically Linked libfips.a

The unified C cryptographic engine

ARCHITECTURE

Implementing PKCS#11 as a Soft-Token

Standardized Abstraction

PKCS#11 offers a uniform interface for cryptographic tokens (Objects, Sessions, Mechanisms).

Soft-Token Design

Replicating key-material isolation and session management entirely within software.

Universal Compatibility

Seamless integration with applications that already consume soft-tokens or HSMs.

Role Separation

Leveraging built-in User and Security Officer roles for API-level access control.

ENGINEERING FRICTION

The "Double Boundary" FFI Sandwich



Inbound FFI (Consumer to Rust)

Safely accepting calls, pointers, and session state from external C applications.

Outbound FFI (Rust to OpenSSL)

Translating sanitized requests down to the C-based cryptographic engine.

State Translation

Converting internal C errors into Rust, mapping to compliant PKCS#11 codes.

IMPLEMENTATION DETAILS

Overcoming API Friction with Rust

PKCS#11 REQUIREMENT	RUST IMPLEMENTATION SOLUTION
HMAC Session Serialization OpenSSL cannot serialize HMAC operations.	Implemented HMAC in Rust using OpenSSL's serializable Hash interface to maintain state.
KBKDF (NIST SP 800-108) OpenSSL provides incomplete specifications.	Reimplemented full derivation spec in Rust using underlying OpenSSL hash primitives.
IPsec Key Derivation Missing entirely from OpenSSL primitives.	Custom implementation built in the Rust layer leveraging the new Rust-based HMAC code.

CERTIFICATION JOURNEY

Defining the Hybrid FIPS Boundary



The Logical Boundary

The entire self-contained dynamic library (.so) serves as the FIPS 140-3 cryptographic boundary.

Unified Integrity

Self-integrity checks span the entire compiled binary at load time.

Power-On Self-Tests (POST)

Known Answer Tests triggered seamlessly upon initialization.

Algorithm Validation

Rust-implemented primitives properly mapped alongside C primitives.

LAB FRICTION POINTS

Certification Experiences

1

ACVP Testing the Rust Layer

Routing Automated Cryptographic Validation Protocol test vectors through Rust algorithms without breaking the harness.

2

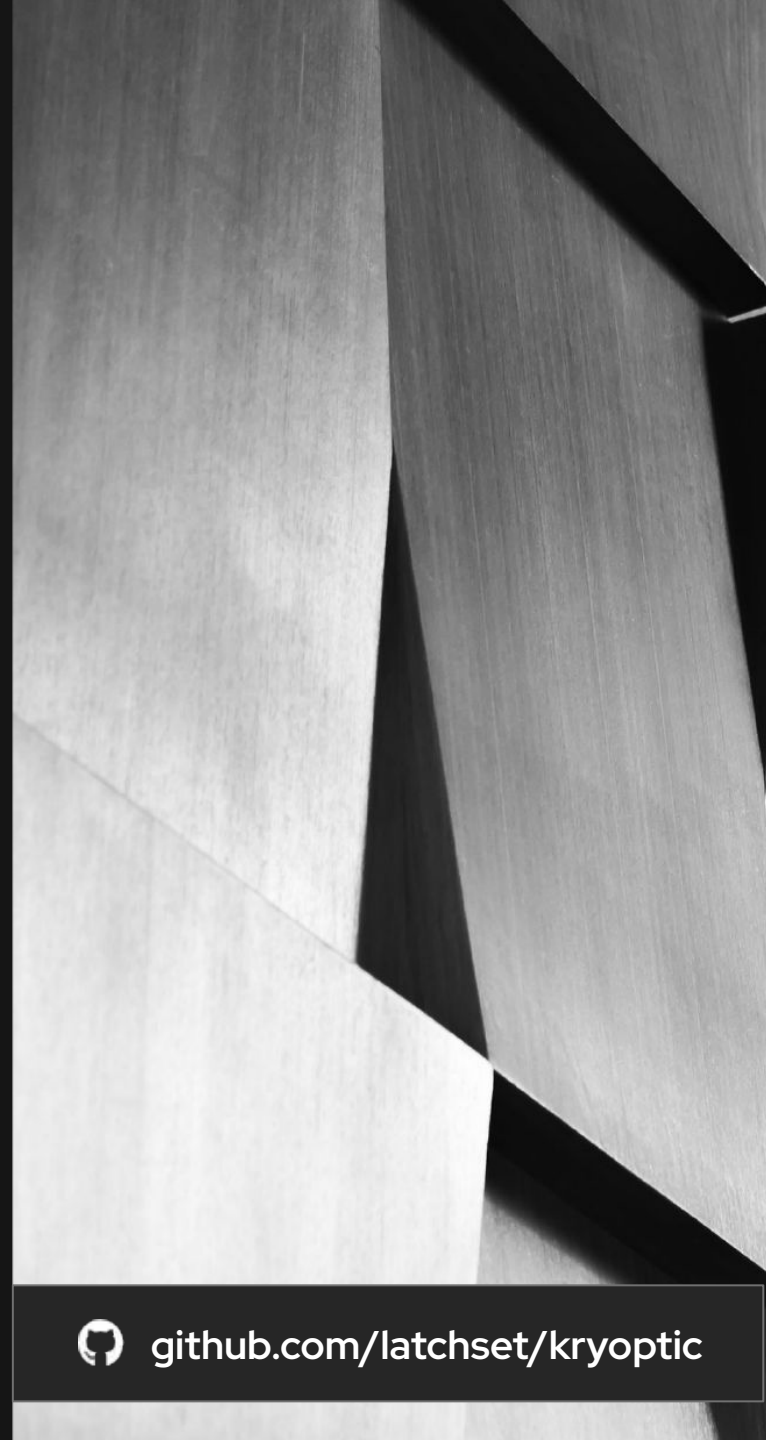
PKCS#11 3.2 FIPS Indicators

- **Explicit Validation:** Standardized interface to raise mode indicators.
- **Queryable State:** Consult library for approved/non-approved status.
- **Granular Tracking:** Explicit status indicators for objects (keys) and sessions.

CONCLUSION

Kryoptic FIPS provider

- **One Module, Multiple Interfaces:** Single dynamic library for OpenSSL & PKCS#11 ecosystems.
- **Minimal Dependencies:** Built for portability and deployment simplicity across Linux.
- **Certified Core:** Merges Rust safety with the proven cryptographic foundation of OpenSSL.
- **Open Source:** Available for review and ecosystem collaboration. github.com/latchset/kryoptic



github.com/latchset/kryoptic

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhat

youtube.com/@redhat

x.com/RedHat

Red Hat