Red Hat
Enterprise Linux

Taking OpenSSL into PKCS #11 World

and Vice Versa

Jakub Jelen
Principal Software Engineer

Red Hat

Jakub Jelen

- Principal Software Engineer, Red Hat
- OpenSC developer and maintainer
  - Open source smart card tools and middleware
- Contributed and improved PKCS #11 support in different projects
  - OpenSSH, libssh, curl, p11-kit, rust-cryptoki, clevis, GnuTLS, ...
- PKCS #11 provider and kryoptic developer
  - Disclaimer: Most of the work done by Simo Sorce

Source:
https://github.com/OpenSC/OpenSC

- PKCS #11
- What's new in PKCS #11 3.2
- OpenSSL and PKCS #11
- pkcs11-provider
- kryoptic

PKCS #11

Cryptographic token interface

PKCS #11: Cryptographic token interface

Cryptoki [crypto key]

Hardware Security Modules and Smart cards

Security: Segregation of cryptographic material

Store Cryptographic information

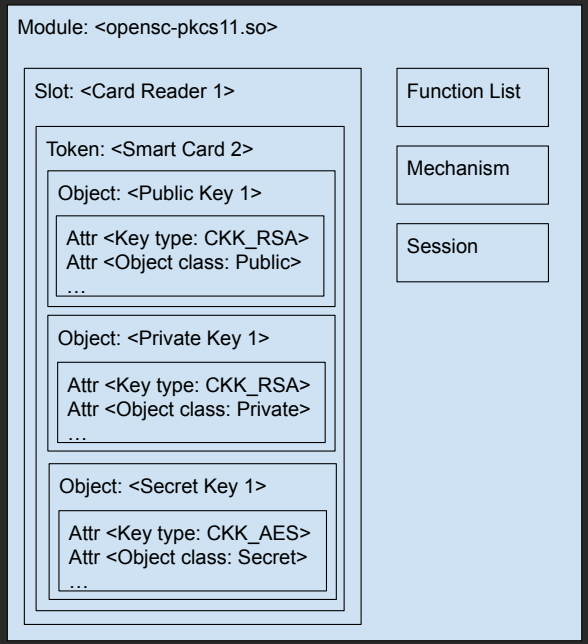
Execute cryptographic operations

Object-based approach

Technology independence

C API

Vocabulary



- Module – software implementing PKCS #11 API
  - GetFunctionList, GetInterfaceList, GetInterface
- Slot + Token: Logical or physical (smart cards)
  - Provide mechanisms and objects, authentication
- Objects: provide attributes: Keys, Certificates, Data …
- Operations:
  - Take Objects, Mechanism and Parameters
  - Result in data or other objects

What's new in PKCS #11 3.2

Post-Quantum Cryptography

- PQC objects:
  - ML-KEM, ML-DSA, SLH-DSA
- PQC Mechanisms
- New API to support PQC operations
  - C_EncapsulateKey(), C_DecapsulateKey()
  - C_VerifySignature*()

Source:
https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.2/csd01/pkcs11-spec-v3.2-csd01.html

Validation API

FIPS service indicators

- Propagation of FIPS validation of HW to user
- Token validation information
  - CKO_VALIDATION object class
- Object validation flags
  - CKA_OBJECT_VALIDATION flag
- Operation validation status (= FIPS Service Indicators)
  - C_GetSessionValidationFlags()

Source:
https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.2/csd01/pkcs11-spec-v3.2-csd01.html

Trust objects

Fine grained trust store

- New object class: CKO_TRUST
  - Maps to existing certificate object
- Different levels of trust/distrust
- For different usage (client, server, code signing, mail, ipsec ...)
- Who likes dealing OpenSSL's CA bundles?

Source:
https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.2/csd01/pkcs11-spec-v3.2-csd01.html

How to use PKCS #11

- Follow the best practices
- Different implementations have different bugs
- A lot of boilerplate needed
  - Initialization, Slot/Token enumeration, Session management, Object enumeration, management and conversion, locking, synchronization
- -> Use OpenSSL pkcs11-provider

PKCS #11 and OpenSSL

Comparison

Relationship

History

Comparison

PKCS #11 x OpenSSL 3.x

C-API: simple Low-Level	x	C-API: rich High Level
Stable, backward compatible	x	Evolving
Slot/Token centric	x	Object centric
Object Attributes	x	Object OSSL_PARAMs
Many implementations	x	Many users
Custom implementations	x	Engines/Providers

PKCS #11 + OpenSSL

The use case

- Familiar high-level OpenSSL API
- Vendors implementing standard PKCS #11 API for HW
- Developers interested in protecting private keys
- Security through separation
  - Break through OpenSSL API: possible
  - Break through OpenSSL, Providers and PKCS #11: ?
  - PKCS#11 is designed to prevent export of private key material

History

- engine_pkcs11 since 2005 (merged into libp11 in 2016)
  - Cumbersome, poor testing and architecture
  - Using engine specific API to load objects
- pks11-provider since 2022
  - Transparent usage with STORE API and PKCS #11 URI
  - SKEY API (Dmitry Belyavskiy later today)
  - PQC mechanisms

Source:
https://github.com/OpenSC/libp11
https://github.com/latchset/pkcs11-provider

pkcs11-provider

A PKCS #11 provider for OpenSSL 3.0+

OpenSSL pkcs11-provider

- Allow applications using OpenSSL to access cryptographic keys and operations on them
- Registers `pkcs11:` URI (RFC 7512) scheme in OpenSSL store
  - Avoids the PKCS#11 token centric approach
  - Returns familiar EVP_PKEY objects
  - The rest of operations same
- Until the user/malware tries to access/export private key material
  - Pkcs11 provider nor the pkcs11 module have access to that!

Source:
https://github.com/latchset/pkcs11-provider

Challenges

Interfaces

- Provider API was undocumented, full of tables
  - Many macros, auto-generated code
- PKCS#11 API requires a lot of boilerplate
  - Generated code
- Recursion: pkcs11 provider in openssl.cnf
  - What if the pkcs11 module is using OpenSSL?
  - SoftHSM crashed or deadlocked because of poor use of libctx

Source:
https://github.com/latchset/pkcs11-provider/blob/main/docs/provider-pkcs11.7.md

Challenges

Early/late initialization

- PKCS#11 initialization is expensive – talks to HW
- What mechanisms we provide when OpenSSL asks?
  - Delay for the first use
- Configuration option pkcs11-module-load-behavior = early

Source:
https://github.com/latchset/pkcs11-provider/blob/main/docs/provider-pkcs11.7.md

Challenges

PKCS #11 implementation bugs

- Software has bugs
- Blocklist functionality with configuration
  - pkcs11-module-quirks = no-operation-state
  - Get/SetOperationState unusable
- Block class of operations
  - pkcs11-module-block-operations = digest, decoder ...
- Prevent PIN locking by repeated attempts
  - Cache PIN digest

Source:
https://github.com/latchset/pkcs11-provider/blob/main/docs/provider-pkcs11.7.md

Testing pkcs11-provider

- The PKCS #11 3.2 specification: almost 500 pages!
- Implementations have bugs
  - All, tokens, pkcs11-provider and OpenSSL
- CI with 3 different software tokens
- Integration tests with different applications: httpd, bind, libssh
- Test coverage for all new algorithms
- Part of OpenSSL CI (external tests)

What's next?

Future work

- PKCS #11 3.2
  - SLH-DSA
  - FIPS status propagation from token:
    - pkcs11-module-assume-fips = yes
  - Trust Objects integration: Provide a Trust store to OpenSSL
- Extend the Symmetric Keys support (Dmitry Belyavskiy later today)
  - ML-KEM, derivation ...
-

kryoptic

A PKCS #11 software token written in Rust using OpenSSL

What is kryoptic?

- Software PKCS #11 implementation
  - using OpenSSL for cryptographic algorithm implementation
- Access OpenSSL primitives through a PKCS #11 API
- Written in rust
- All modern algorithms (including PQC)
- Useful (not only) for testing pkcs11-provider
- Side product: ossl crate: lightweight OpenSSL 3 native bindings

Source:
https://github.com/latchset/kryoptic/
https://crates.io/crates/ossl

Why rust?

Its cool!

- Memory safety:
  - Bounds checks, thread safety at build time
  - Automatic memory management, but no GC
  - No runtime performance penalties
- Easy interfacing with C code
- A lot of refactors
  - Helpful compiler makes it safe
- Steep learning curve

Interfacing C code from Rust

Memory safety limits

- PKCS #11: FFI from application to Rust code
  - Convert inputs using unsafe { }
  - bindgen crate

Kryoptic: Token functionality: slots, sessions, mechanisms, locking, threading and more

- FFI from Rust to OpenSSL's code
  - Call functions using unsafe {}
  - ossl crate

OpenSSL inside of OpenSSL

For all the operations!

- Configure openssl.cnf with propq to use pkcs11 provider: -propquery="?provider=pkcs11"
- Configure pkcs11 provider to use kryoptic
- Offload the cryptographic operation from OpenSSL to OpenSSL
  - Test coverage

Source: Gemini

Kryoptic as a FIPS module

Two-headed monster

- Kryoptic can be statically linked with OpenSSL libfips.a
  - Few OpenSSL API tweaks
- Single static FIPS library with 2 APIs
  - Provider API: Usable in OpenSSL
  - PKCS #11: usable in NSS, GnuTLS, ...

Challenges

- PKCS #11 allows a lot of flexibility in some mechanisms
  - NIST SP800-108 KBKDF – IV lengths, odd combinations
- Static linking with FIPS provider
  - Different API for DigestSign API
  - Manual invocation of constructors, callbacks, tables
- Memory safety ends on FFI (`unsafe {}`)
  - Rust can not guarantee what happens behind that
  - OpenSSL API expectations regarding the buffer bounds

  -> test coverage needed

Summary

PKCS #11: an API to access HSM and smart cards

The latest release PKCS #11 3.2 comes with PQC

Use pkcs11-provider to do so through OpenSSL

Testing? Replace the HW with kryoptic

New project? Use rust! We have ossl bindings!

Source:
https://crates.io/crates/ossl

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


youtube.com/user/RedHatVideos


facebook.com/redhatinc


x.com/RedHat

Red Hat