Beyond Algorithms: Exploiting Platform Cryptography with OpenSSL Providers

OpenSSL Conference 2026, Prague

Holger Dengler (dengler@linux.ibm.com)

IBM Deutschland R&D GmbH

Outline

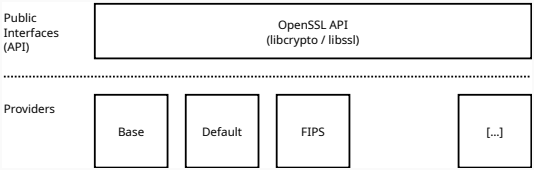
Provider Basics

Platform Cryptographic Features

hbkzpc Provider

Provider Basics

OpenSSL Providers



Providers: Purpose

For what is it used?

- Algorithms
  - OQS Provider (new algs)
  - PKCS11 Provider (adopt APIs)
- Key storage
  - PKCS11 Provider (PKCS#11 URI)
- File encodings
  - DER/PEM encoding (new key storage URI)
  - Key encodings

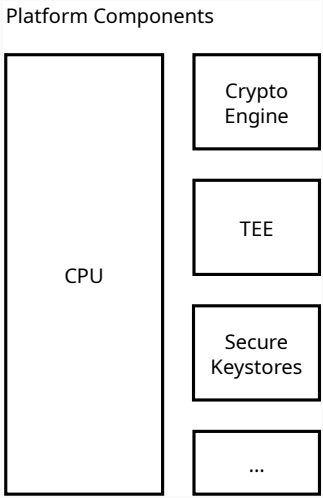
Provider:
Functional (dispatch)
- en-/decoder
-store-loader
- keymgmt (sym/asym)
- cipher (sym/asym)
- mac/signature
- keyexch/kdf/kem
- digest
- rand
- ...

Provider Configuration
- Name
- Version
- Module
- ...

Platform Cryptographic Features

Platform: Cryptographic Components

- CPU
  - crypto instructions
- Crypto Engines
  - accelerators
  - offloading
  - protected operation
- Keystores
  - protected storage
- ...



Platform: Hardware-backed Keys

- clear key never readable in operating system/application
- key references/handles or wrapped keys
- implementation in separate hardware, firmware or CPU feature
- cryptographic operations: implicit clear key dereference or unwrap

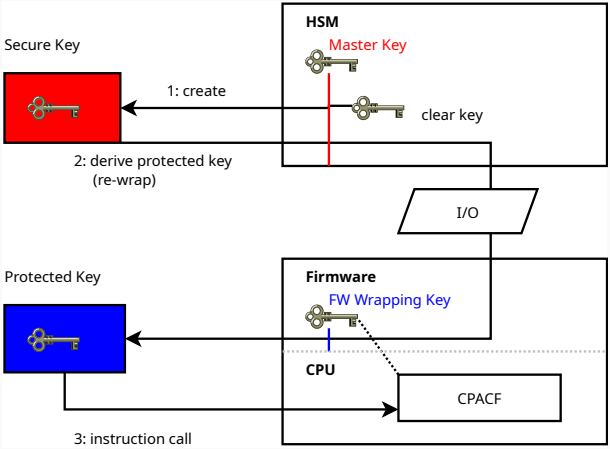
Cryptographic Acceleration on IBM Z

- Central Processor Assist for Cryptographic Function (CPACF)
  - instruction set for cryptographic operations
  - long running instructions, multiple cipher blocks per call
  - instructions are interruptable and resumable
  - e.g. instruction KMC -> AES-128/192/256-CBC (w/ clear key or protected key)

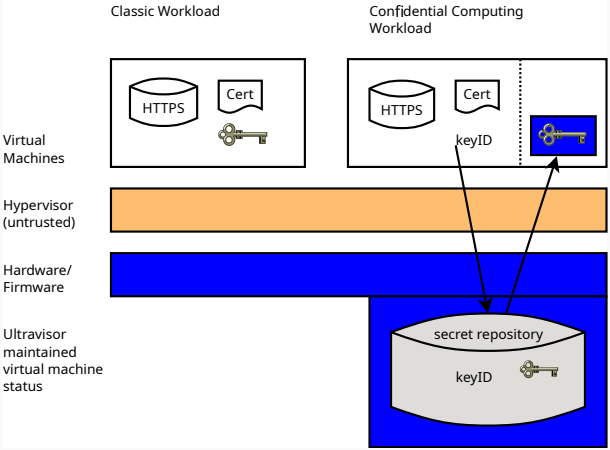
Hardware-backed Keys on IBM Z

- protected key
  - derived from key origins (persistent)
    - secure key (HSM)
    - retrievable secret ID (Confidential Computing)
    - clear key (Development/Test)
  - derivation: re-wrap with firmware wrapping key (volatile)
    - FW wrapping key specific to virtual machine (partition, guest)
  - in main memory: only wrapped key
  - related clear key is never exposed to platform hypervisor, operating system or user-space
    - application: instruction KMC is called with the wrapped key (protected key)
    - firmware: implicit key unwrap
    - firmware: perform AES-CBC algorithm

Hardware-backed Keys on IBM Z (Secure Key Example)



Hardware-backed Keys on IBM Z (Retrievable Secret Example)



hbkzpc Provider

Motivation & Goals

- Motivation
  - expanding the usage base of protected-key cryptography
  - replace the proprietary API for protected-key cryptography
- Goals
  - exploit protected-key cryptography via established crypto API
  - minimize the effort required for adoption
  - make it accessible to a broad set of software packages
  - ideal: only configuration and protected-key origin files

Design Decisions

- protected-key cryptography via OpenSSL API
- key origins as files (DER/PEM)
- URI for key origins
- internally (re)use existing zpc API
- tooling support for protected-key origins

hbkzpc Provider: Implementation

- current (provider)
  - store-loader (for the hbkzpc URI)
  - decoder (for PEM/DER encoded files)
  - keymgmt (for loading keys)
  - signature (for ECC key origins)
- current (key management tool)
  - compose key origins (private-key-only and key-pair)
  - output-format: URI, DER, PEM
- upcoming
  - encoder
  - cipher (AES)
  - mac (HMAC-SHA2)
  - HSM-based key origins
  - PKCS#11 interface

hbkzpc Provider: URI

URI

- hex-strings for BLOBs
- for Ultravisor (UV) key origins only: public key

URI

```
hbkzpc:origin-alg=<alg>;origin-type=<type>;\
       origin-blob=<hex>;origin-pubkey=<hex>;\
       comment=<string>
```

hbkzpc Provider: DER and PEM encoding

File encodings:

- DER: ASN.1 sequence of version and URI
- PEM: base64 encoding of the DER

DER

```
SEQUENCE
cont [ 0 ]
VISIBLESTRING     :HBKZPC Provider URI v1.0
cont [ 1 ]
UTF8STRING        :hbkzpc:origin-type=[...]
```

PEM

```
-----BEGIN HARDWARE BACKED KEY ZPC-----
<base64 data>
-----END HARDWARE BACKED KEY ZPC-----
```

hbkzpc Provider: Configuration

In the OpenSSL Configuration:

- module name
- activate
- Fedora/RHEL: minimal configuration drop-in

Configuration Drop-In

```
[provider_sect]
hbkzpc = hbkzpc_sect

[hbkzpc_sect]
module = zpcprovider.so
activate = 1
```

hbkzpc Provider: Usage

Sign Message (with key-origin)

```
openssl dgst -sha256 -sign p-256-hbkzpc.pem \
   -out message.sig message.txt
```

Verify Message (with key-origin)

```
openssl dgst -sha256 -verify p-256-hbkzpc.pem \
   -signature message.sig message.txt
```

Verify Message (with public key)

```
openssl dgst -sha256 -verify p-256-pub.pem \
   -signature message.sig message.txt
```

hbkzpc Provider: Summary

- Provider concept enables us to meet all our goals!
- it is that simple: configure, replace key-files, and that's it!
- on Fedora/RHEL: configuration at package install
- available since version 2.0 on GitHub: [1]

Conclusion

- Provider API makes it easy to exploit Platform Crypto-Capabilities
- take advantage of OpenSSL's widespread adoption
- handle new key formats via Store / Decoder / Encoder / Key-Management
- minimize the effort for maintenance

References

libzpc Repository. [Online]. Available: https://github.com/opencryptoki/libzpc.git

Beyond Algorithms: Exploiting Platform Cryptography with OpenSSL Providers

OpenSSL Conference 2026, Prague

Holger Dengler (dengler@linux.ibm.com)

IBM Deutschland R&D GmbH