Crypto Acceleration on FreeBSD

Philip Paeps
philip@FreeBSD.org

The FreeBSD Project

BSDCan 2009 — Ottawa, Canada
8 May 2009

Outline

Outline

In The Beginning...

- Developed for OpenBSD by Angelos D. Keromytis
- Consistent software and hardware interface
- Fairly modular and extendable design
- Ported to FreeBSD by Sam Leffler in 2002
- Originally particularly intended for IPSEC
- Very little of the original code remains

List of Components

- Kernel "library" for crypto operations
- Generic software crypto device
- Support for acceleration hardware
- Interface to userland for acceleration

Kernel “Library” for Crypto Operations

- “OpenSSL of the kernel”
- Reduces code duplication like a library
- Fairly self-contained and maintainable
- Most functionality is in `<opencrypto/cryptodev.h>`

Software Crypto Device

- Implemented as a “transformation” system
- Supports most relevant crypto transformations
  - MD5, SHA1, Rijndael, Camellia, . . .
  - Very flexible and easy to extend
- Behaves exactly like a “hardware” device

Acceleration Hardware

- Crypto acceleration hardware widely available
- Currently mostly useful for “slow” system
- Framework automatically takes advantage of hardware
- Uses software if no hardware is present

Interface to Userland

- As simple as `ioctl` on `/dev/crypto`
- Asynchronous session-oriented interface
- Mainly used by the OpenSSL "cryptodev" `ENGINE`
- Not on by default — some serious drawbacks!

Consumers in the Kernel

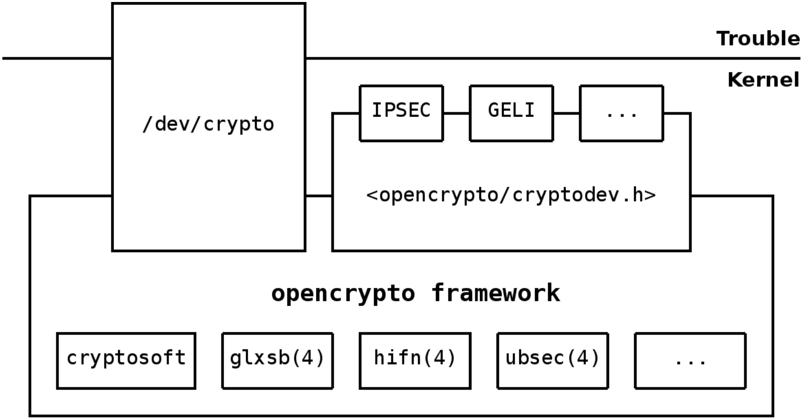
- IPSEC
- Block devices (GELI)
- Wireless (IEEE 802.11)
- ZFS
- GSSAPI

OpenSSL Engine

- Limited adoption due to default disabled
- This is perhaps a good thing
- Patches floating around for many applications
- Often $<$ 10 lines of code needed

Outline

Architectural Overview



Modes of Operation

Session-based mode

- Information cached per driver (perhaps in hardware)
- No need to repeat initialization for every operation
- Multiple operations can be chained together

Sessionless mode

- Used mainly for keying operations or hashing
- Input and output parameters passed in with request
- No support for multiple operations

Asynchronous Interface

- Both modes of operation are asynchronous
- Consumers are not necessarily processes
- Drivers not able to `sleep(9)`
- Callback mechanism for status and errors

Kernelspace Usage

- `#include <opencrypto/cryptodev.h>`
- Initialize session parameters once
- Multiple operations can be chained together
- Framework takes care of the rest

Outline

Supported Devices

- `glxsb(4)` — AMD Geode
- `hifn(4)` — Hifn
- `padlock(4)` — VIA Padlock
- `safe(4)` — SafeNet
- `ubsec(4)` — Broadcom/Bluesteel

Driver-side API

- Drivers register algorithms they support with the framework
- Callbacks for session management and for dispatching work
- Sessions are managed by the drivers
- Currently no real “prioritization” support
- Mostly compatible with OpenBSD for now
- No support for cyphertext stealing (*point to Doug*)

Kernel-side Issues

- Sessions are managed by device drivers
- No (real) way to migrate sessions
- Very minimal support for prioritization
- Limited flow-control opportunities

Trends in Hardware

- Acceleration hardware becoming faster
- Moving from slow(ish) PCI bus to integrated
- Becoming more and more like co-processors

Further Problems in Userland

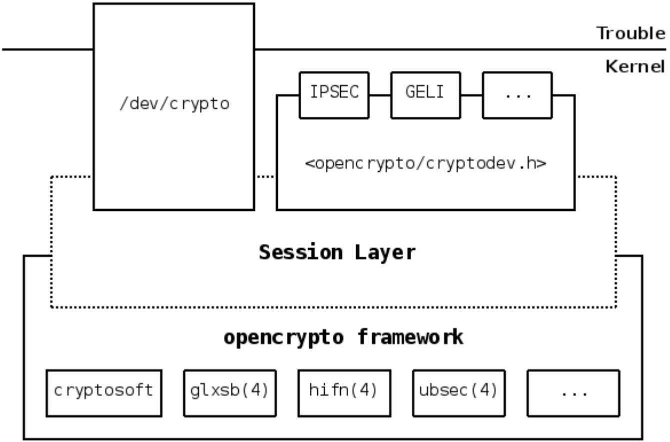
- OpenSSL `ENGINE`s are “all or nothing”
- Context switching often very undesirable
- No heuristic for deciding if acceleration makes sense
- On a fast machine, software is often fastest

Outline

Session-Management Layer

- Add support for migrating sessions between drivers
- Enable load-balancing across multiple devices
- Heuristics to determine if software is faster
- Tell userspace (OpenSSL) if no hardware is available

Architectural Overview



Improve Parallelism

- Hardware is evolving towards multiple execution blocks
- Provide a method for flow-control towards hardware
- Add support to pin sessions to a single CPU

More Hardware Support

- Currently, mostly low-end embedded hardware supported
- Some really sexy high-end devices are available
- Often a small matter of programming and access to hardware

Questions? Comments?