Crypto Engine on Raspberry Pi 5

System Understanding, Benchmarking and Demo

Group
Jack, Jesse, Omar, Thu Ta

ECE 4301

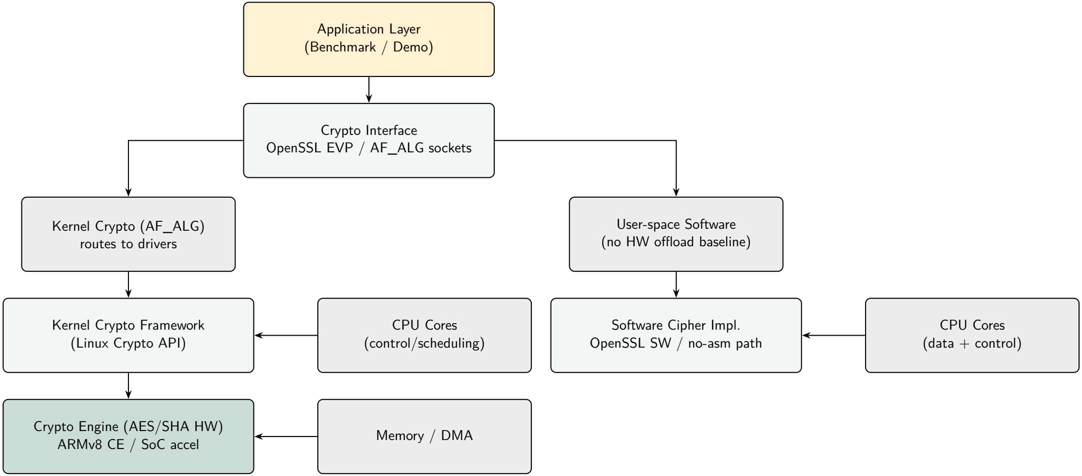
Oct 6 2025

Agenda

What is a Crypto Engine?

- Dedicated hardware for AES, SHA, etc.
- Reduces CPU load and latency; boosts throughput.
- Exposed via Linux Crypto API (AF_ALG), OpenSSL EVP/engines, and `/proc/crypto`.
- Used in TLS/SSH, disk encryption, secure storage.

Crypto Engine System Overview



Two paths benchmarked: **AF_ALG (kernel→HW engine)** vs. **Software-only** (no HW offload).

Pi 5 Placement (Conceptual)

User Space

Application (AES file encrypt, TLS / benchmark)
OpenSSL EVP or AF_ALG sockets API

Kernel Space (HW path)

Linux Crypto API & AF_ALG → drivers (`aes-ce`, `sha256-ce`)

SoC / Hardware

ARMv8 Crypto Extensions / SoC accelerator (AES/SHA)
Cortex-A76 CPU coordinating with Memory/DMA

OS Interaction (Linux Path)

Discovery / Capabilities

```
# HW-backed algorithms exposed by kernel:
grep -E 'name.*(aes|sha1|sha256)' -A3 /proc/crypto

# Optional: AF_ALG via OpenSSL engine (if present) or AF_ALG sockets
openssl engine -t -c    # may or may not list 'afalg'
```

Layers

- **AF_ALG path (HW):** App → AF_ALG/EVP → Kernel Crypto → HW engine.
- **Software path:** App → OpenSSL SW cipher (no HW) → CPU only.

Scheduling

Kernel schedules AF_ALG ops; CPU overlaps control while HW engine moves blocks via DMA.

Example: AES-CBC Benchmark Calls

Hardware path (AF_ALG)

```
# Either via OpenSSL engine (if available) ...
openssl speed -elapsed -engine afalg -evp aes-256-cbc

# ...or your AF_ALG socket/kcapi harness (as used for the plots).
```

Software-only baseline

```
# OpenSSL software cipher build / path (no HW offload)
openssl speed -elapsed -evp aes-256-cbc   # built w/ SW-only for baseline
```

What we varied

- Total sizes: 64/128/256 MB   Chunk sizes: 4–1024 KB
- Operations: encrypt & decrypt; record MB/s (and CPU% when sampled)

Benchmark Plan & Reproducibility

Environment setup (Pi 5)

```
python3 -m venv venv
source venv/bin/activate
sudo apt install -y build-essential linux-headers-$(uname -r) \
  python3 python3-pip python3-matplotlib python3-pandas
pip3 install pandas matplotlib
```

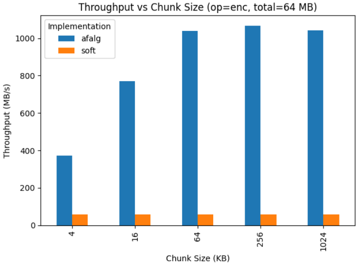
Run benchmark + plots

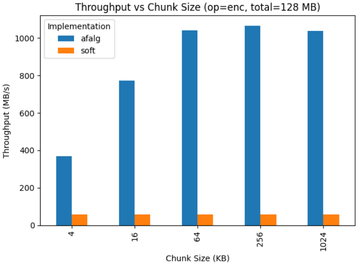
```
./run_bench.sh
python3 plot.py
```

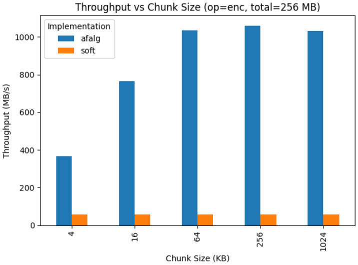
What it measures

- Encrypts/decrypts random buffers; chunk sizes 4 KB–1 MB; totals 64/128/256 MB
- Metrics: **Throughput (MB/s)** and **CPU time (ms)**
- Saves to `results.csv`; generates throughput and CPU-efficiency plots

AES Encryption Throughput: `afalg` vs `soft`

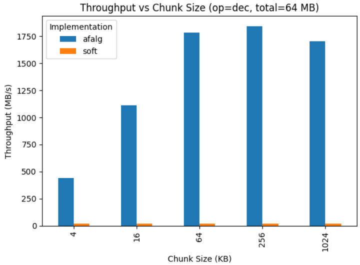


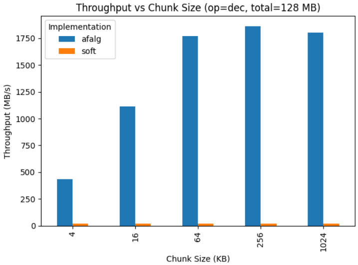


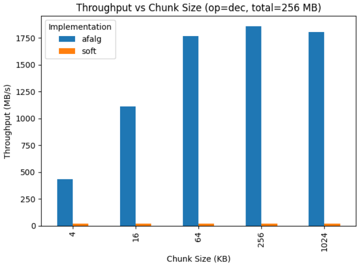


Trend: Throughput increases with chunk size and plateaus ≈1.0–1.1 GB/s (afalg). Software-only remains ≈40–60 MB/s across sizes.

AES Decryption Throughput: `afalg` vs `soft`

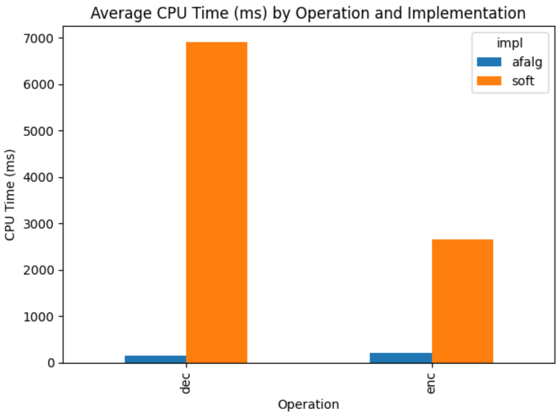






Trend: Decrypt scales similarly; afalg peaks ≈1.7–1.9 GB/s with large chunks. Gap vs. software-only is persistent across totals.

CPU Time (ms): afalg vs soft



Average CPU time shows a clear efficiency gap: **afalg** consumes only a few hundred ms per run, while **soft** takes several seconds. Roughly $\sim$10$\times$ lower CPU time for *enc* and $\sim$40$\times$ lower for *dec*, matching the throughput advantage.

Demo (Completed): Setup & Method

Virtual env & deps

```
python3 -m venv venv
source venv/bin/activate
sudo apt install -y build-essential linux-headers-$(uname -r) \
  python3 python3-pip python3-matplotlib python3-pandas
pip3 install pandas matplotlib
```

Run demo

```
./run_bench.sh
python3 plot.py
```

Evaluation

- Compare AF_ALG (kernel→HW engine) vs software AES-128-CBC
- Chunk sizes: 4 KB–1 MB; Totals: 64/128/256 MB; enc & dec
- Outputs: `results.csv` + plots used in these slides

Demo (Completed): Key Findings

- **afalg** saturates at ∼1.0–1.1 GB/s (enc) and ∼1.7–1.9 GB/s (dec).
- **soft** remains ∼40–60 MB/s; CPU time several seconds vs few hundred ms on afalg.
- Larger chunks reduce per-call overhead; knee near 64–256 KB.
- Crypto engine frees CPU cycles for other work.

Summary

- **How it works:** Apps use OpenSSL EVP or AF_ALG sockets; the kernel routes AES/SHA requests to ARMv8 Crypto Extensions (HW engine) or to a software-only path.
- **What we benchmarked:** Encrypt/decrypt random buffers across totals of 64/128/256 MB with chunk sizes 4 KB–1 MB; recorded **throughput (MB/s)** and **CPU time (ms)**; saved to `results.csv` and plotted.
- **Results (Pi 5):**
  - **Throughput:** `afalg` ≈1.0–1.1 GB/s (enc), ≈1.7–1.9 GB/s (dec); `soft` ≈40–60 MB/s.
  - **CPU time:** `afalg` ∼0.15–0.25 s per run vs ∼2.6–6.9 s for `soft`.
  - Larger chunks (256 KB) approach peak throughput and best CPU efficiency.

References & Resources

- Raspberry Pi Docs — `raspberrypi.com/documentation`
- BCM2712 SoC Datasheet — `datasheets.raspberrypi.com`
- ARMv8 Crypto Extensions — `developer.arm.com`
- OpenSSL Documentation — `openssl.org/docs`
- Linux Crypto API Guide — `kernel.org/doc/html/latest/crypto/`