

# Linux Crypto API is dead!

Long live Linux Crypto API v2?

Ignat Korchagin

# \$ whoami

- Linux at Citadel
- Systems security and performance
- Low-level programming
- Co-maintainer of the asymmetric keys

# Linux (userspace) Crypto API is dead!

## User Space Interface

### Introduction

AF\_ALG provides unprivileged userspace programs access to arbitrary hash, symmetric cipher, AEAD, and RNG algorithms that are implemented in kernel-mode code.

AF\_ALG is insecure and is deprecated. Originally added to the kernel in 2010, most kernel developers now consider it to be a mistake. Support for hardware accelerators, which was the original purpose of AF\_ALG, has been removed.

AF\_ALG continues to be supported only for backwards compatibility.

Starting in Linux v7.3, the set of algorithms supported by AF\_ALG is limited by default. See [/proc/sys/crypto/af\\_alg\\_restrict](#).

On systems where no programs using AF\_ALG remain, the support for it should be disabled entirely by setting `/proc/sys/crypto/af_alg_restrict` to 2 or by disabling `CONFIG_CRYPTO_USER_API_*` in the kernel configuration.

### Deprecation

AF\_ALG was originally intended to provide userspace programs access to crypto accelerators that they wouldn't otherwise have access to.

However, that capability turned out to not be useful on very many systems. More significantly, the actual implementation exposes a vastly greater amount of functionality than that. It actually provides access to all software algorithms.

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- Steady stream of CVEs (“copy fail” and friends)

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- Steady stream of CVEs (“copy fail” and friends)
- Just exposes “pure math” software algorithms
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- Steady stream of CVEs (“copy fail” and friends)
- Just exposes “pure math” software algorithms
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Steady stream of CVEs

- With new LLMs who doesn't?



# Steady stream of CVEs

- With new LLMs who doesn't?
- It is not only the AF\_ALG interface. What about “Dirty Frag”?
- CVE-2026-43284 (IPsec ESP/XFRM)
- CVE-2026-43500 (RxRPC)
- CVE-2026-46300 (ESP-in-TCP)



# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- Steady stream of CVEs (“copy fail” and friends)
- Just exposes “pure math” software algorithms
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Linux (userspace) Crypto API is dead!

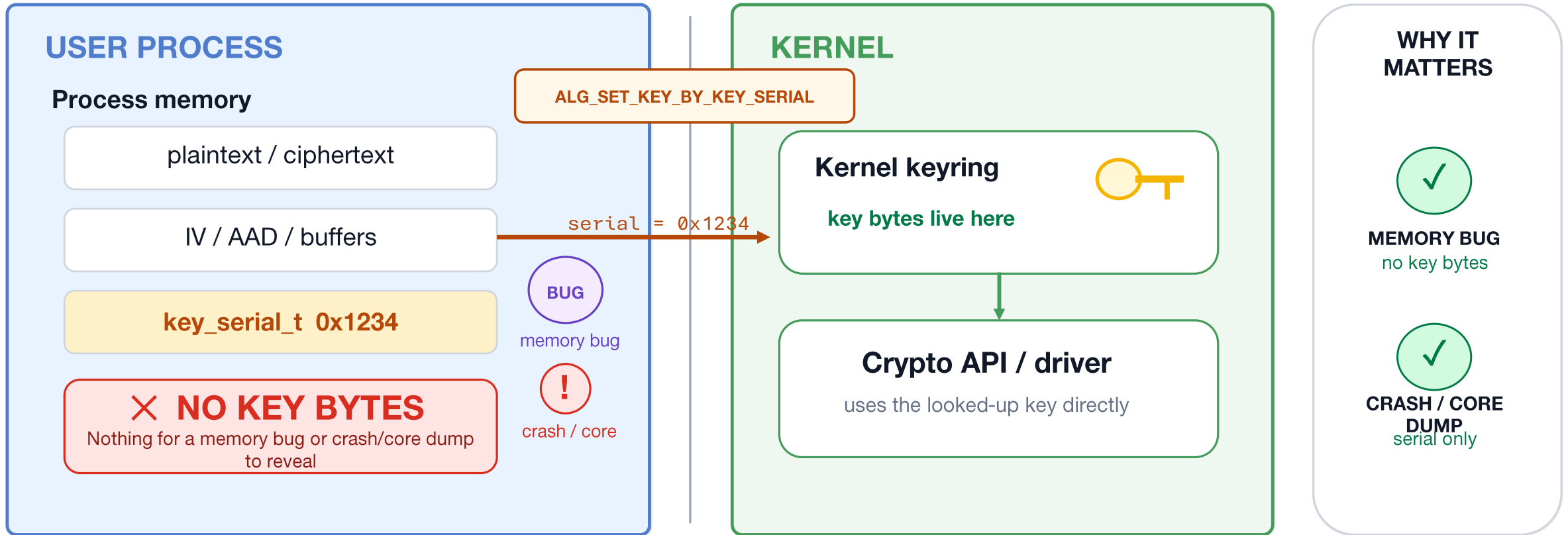
- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- ~~Steady stream of CVEs (“copy fail” and friends)~~
- Just exposes “pure math” software algorithms
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Not just “pure math”

- Math + key memory isolation!
- via `ALG_SET_KEY_BY_KEY_SERIAL`

# Math + memory isolation!

AF\_ALG + ALG\_SET\_KEY\_BY\_KEY\_SERIAL keeps key bytes out of the consuming process address space



```
setsockopt(fd, SOL_ALG, ALG_SET_KEY_BY_KEY_SERIAL, &serial, sizeof(serial));
```

Source: Linux kernel documentation — User Space Interface / ALG\_SET\_KEY\_BY\_KEY\_SERIAL

Math + memory isolation!



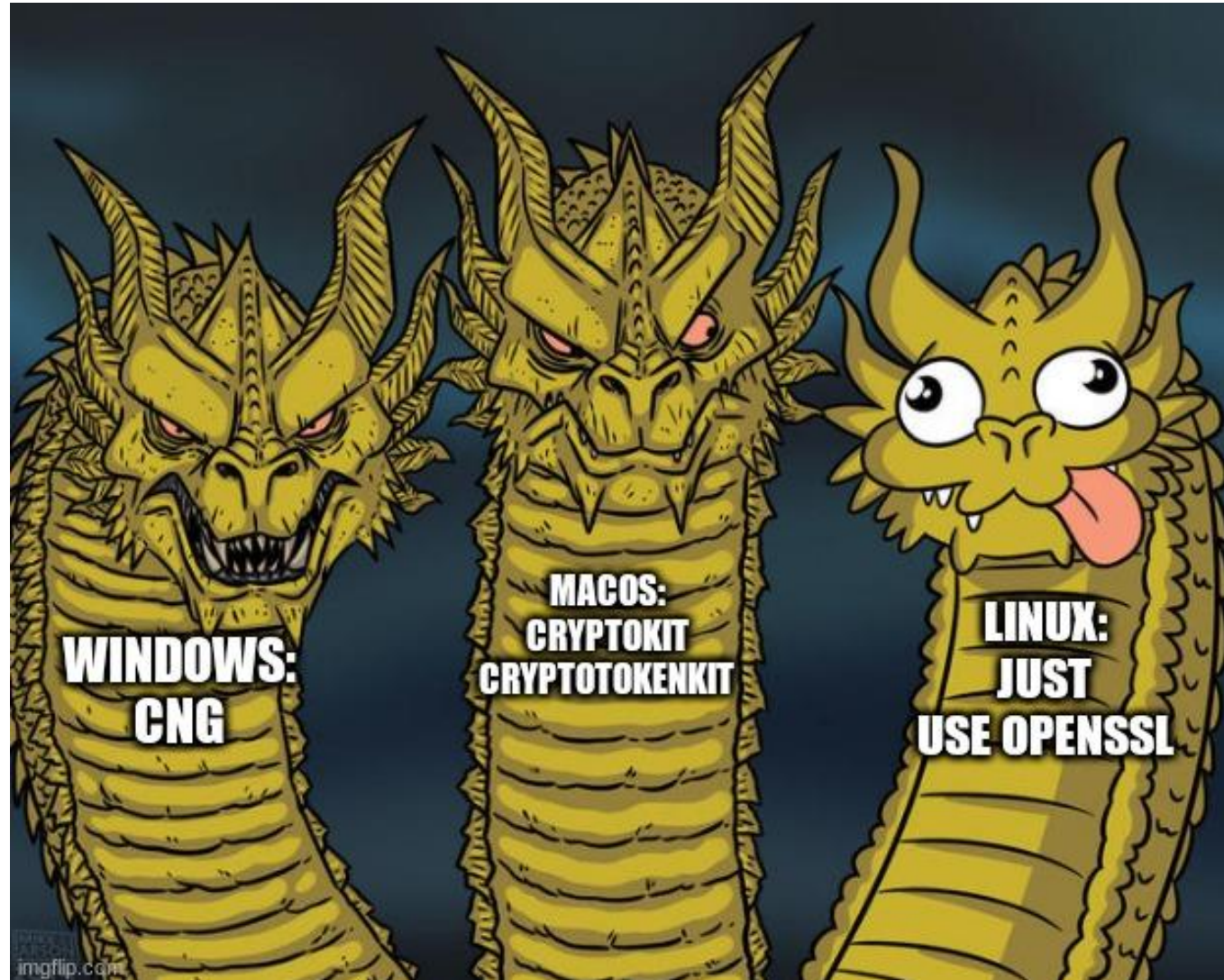
Math + memory isolation!



# Not just “pure math”

- Math + key memory isolation!
  - via ALG\_SET\_KEY\_BY\_KEY\_SERIAL
- Common OS provided Crypto API interface
  - Crypto accelerators
  - Pluggable modules

# OS provided Crypto API interface



# OS provided Crypto API interface



# Crypto accelerators bespoke APIs

- Rambus CryptoManager Hub (v5, 17 Sept 2026)
  - 1 device file, 44 ioctls
- NXP EdgeLock Secure Enclave (v51, 14 Sept 2026)
  - multiple device nodes (/dev/<se>\_muN\_chN), 5 ioctls
- Altera/Intel SoCFPGA FCS (v4, 4 Sept 2026)
  - 1 device file, 1 ioctl

# Crypto accelerators bespoke APIs

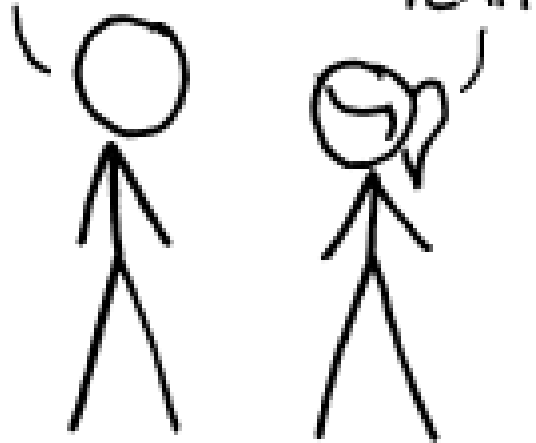


# We need a standard pluggable API

HOW STANDARDS PROLIFERATE:  
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:  
THERE ARE  
14 COMPETING  
STANDARDS.

14?! RIDICULOUS!  
WE NEED TO DEVELOP  
ONE UNIVERSAL STANDARD  
THAT COVERS EVERYONE'S  
USE CASES.



SOON:

SITUATION:  
THERE ARE  
15 COMPETING  
STANDARDS.

# Not just “pure math”

- Math + key memory isolation!
  - via `ALG_SET_KEY_BY_KEY_SERIAL`
- Common OS provided Crypto API interface
  - Crypto accelerators
  - Pluggable modules
- Increased code reuse
  - Useful for constrained embedded systems where every byte counts

# Cost of statically linking a userspace crypto library (x86)

| Library           | Bytes     |
|-------------------|-----------|
| mbedTLS 2.28.8    | 110,312   |
| OpenSSL 4.2.0-dev | 2,341,103 |

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- ~~Steady stream of CVEs (“copy fail” and friends)~~
- Just exposes “pure math” software algorithms
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- ~~Steady stream of CVEs (“copy fail” and friends)~~
- ~~Just exposes “pure math” software algorithms~~
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

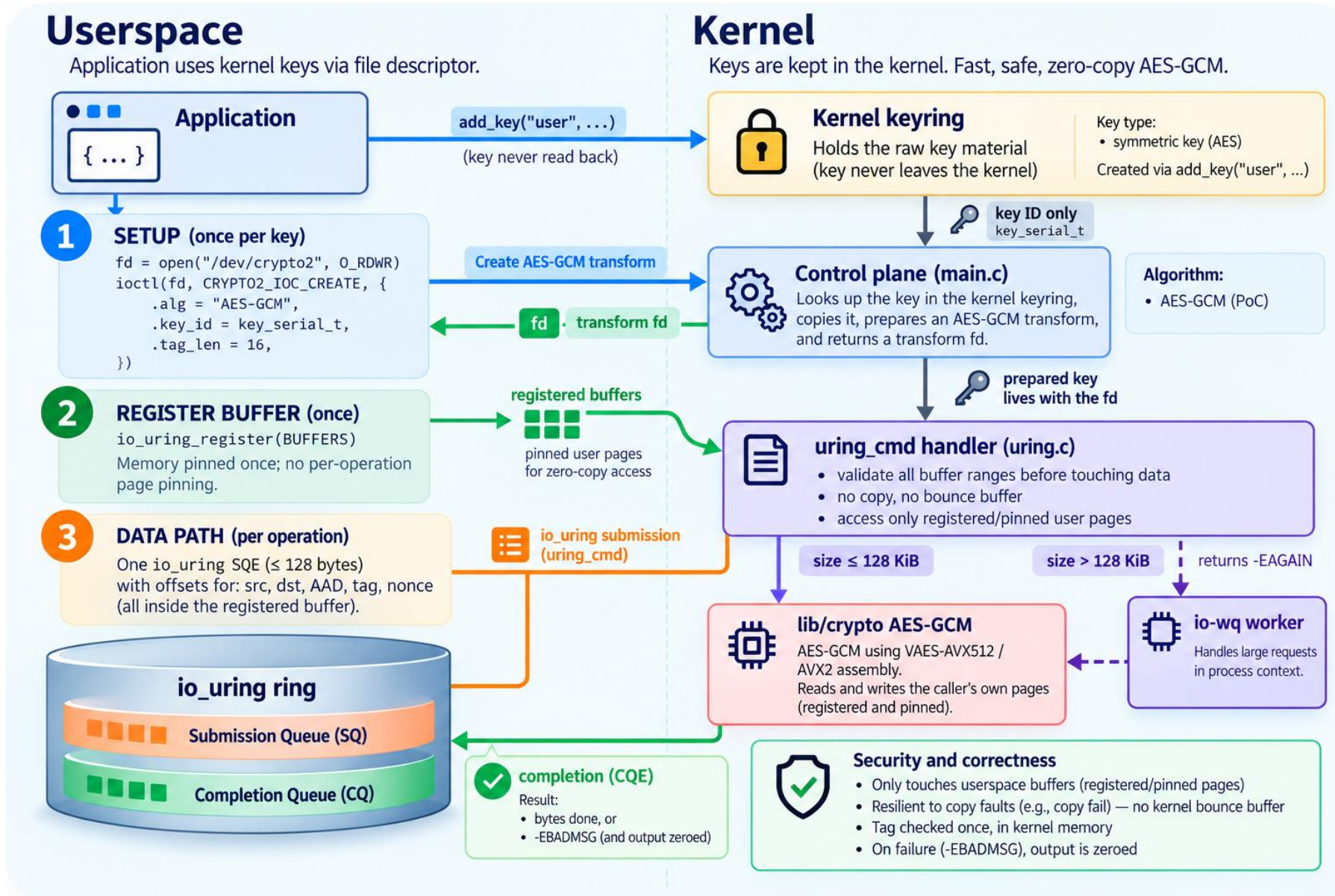


# Long Live Crypto API v2?

Code is cheap \*

\* if your employer pays for tokens

# Crypto API v2 using io\_uring!



# Crypto API v2 using io\_uring!

| Message | Conditions                          | crypto2                 | OpenSSL                 |
|---------|-------------------------------------|-------------------------|-------------------------|
| 64 B    | one at a time                       | 404 ns (0.16 GB/s)      | 77 ns (0.83 GB/s)       |
| 1 KiB   | one at a time                       | 458 ns (2.2 GB/s)       | 142 ns (7.2 GB/s)       |
| 1 KiB   | 64 in flight                        | 254 ns (4.0 GB/s)       | not measured            |
| 16 KiB  | 64 in flight                        | 1.1 $\mu$ s (14.8 GB/s) | 1.0 $\mu$ s (16.1 GB/s) |
| 64 KiB  | 64 in flight                        | 3.9 $\mu$ s (16.8 GB/s) | 3.7 $\mu$ s (17.6 GB/s) |
| 1 MiB   | in flight, no copy, hugepage buffer | 59 $\mu$ s (17.8 GB/s)  | 59 $\mu$ s (17.9 GB/s)  |

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - Slow, so negates the use of crypto accelerators
  - Exposes arbitrary composition of crypto algorithms
- ~~Steady stream of CVEs (“copy fail” and friends)~~
- ~~Just exposes “pure math” software algorithms~~
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Linux (userspace) Crypto API is dead!

- Somewhat clunky bespoke interface: AF\_ALG sockets
  - ~~• Slow, so negates the use of crypto accelerators~~
  - Exposes arbitrary composition of crypto algorithms
- ~~• Steady stream of CVEs (“copy fail” and friends)~~
- ~~• Just exposes “pure math” software algorithms~~
  - Accelerators are buggy
  - Use OpenSSL (or other libs) instead

# Other considerations

- Expose lib/crypto as vDSO
  - Only good for embedded use case
  - No memory isolation
  - Does not solve accelerator problem
  - Per-process memory inflation, even if not using crypto

# Other considerations

- Expose lib/crypto as vDSO
  - Only good for embedded use case
  - No memory isolation
  - Does not solve accelerator problem
  - Per-process memory inflation, even if not using crypto
- Linux PEM (pluggable encryption modules, like Linux PAM)
  - No memory isolation problem by default
  - Needs widespread distro adoption
  - Accelerators still need bespoke kernel/userspace interfaces

# Other considerations

- Expose lib/crypto as vDSO
  - Only good for embedded use case
  - No memory isolation
  - Does not solve accelerator problem
  - Per-process memory inflation, even if not using crypto
- Linux PEM (pluggable encryption modules, like Linux PAM)
  - No memory isolation problem by default
  - Needs widespread distro adoption
  - Accelerators still need bespoke kernel/userspace interfaces
- Device files, sysfs etc
  - No zero-copy
  - May re-introduce “Copy Fail”

Questions, comments?

