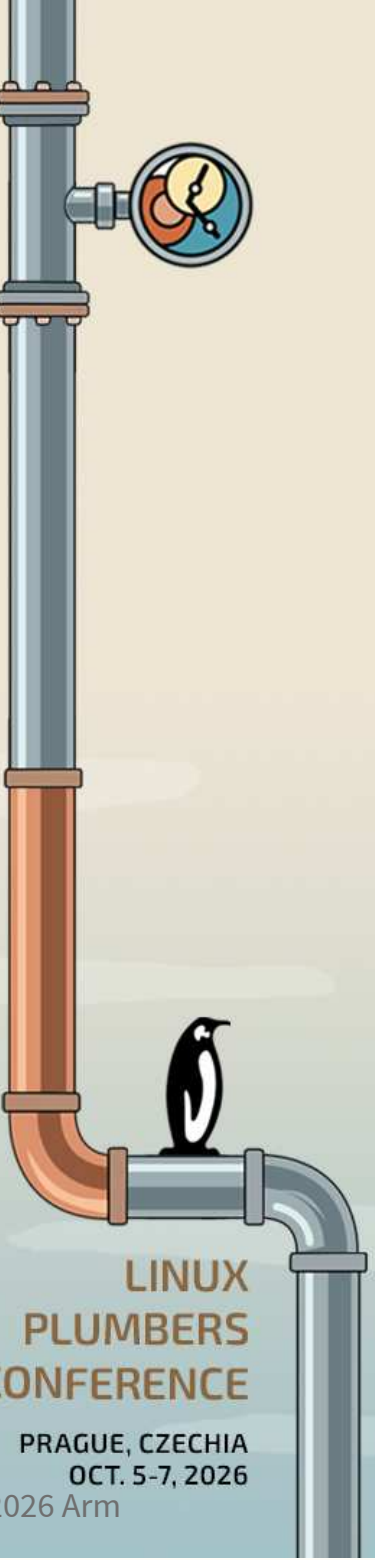


# Defining the pkeys ABI

BoF

Kevin Brodsky

**arm**



LINUX  
PLUMBERS  
CONFERENCE

PRAGUE, CZECHIA  
OCT. 5-7, 2026

© 2026 Arm

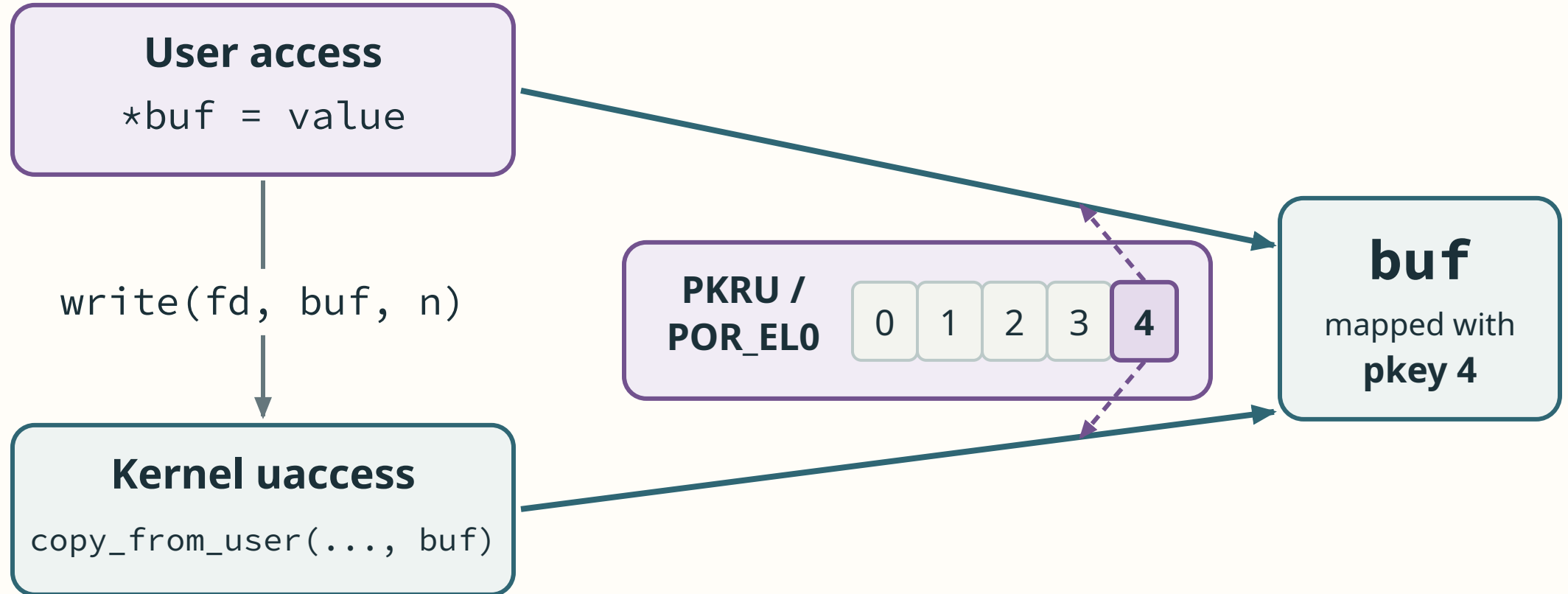
*“We really need to sit down and actually define a proper programming model first instead of trying to duct tape the current ill defined mess forever.”*

— Thomas Gleixner, Re: [PATCH v7 3/4] rseq: Make rseq work with protection keys [2025]

---

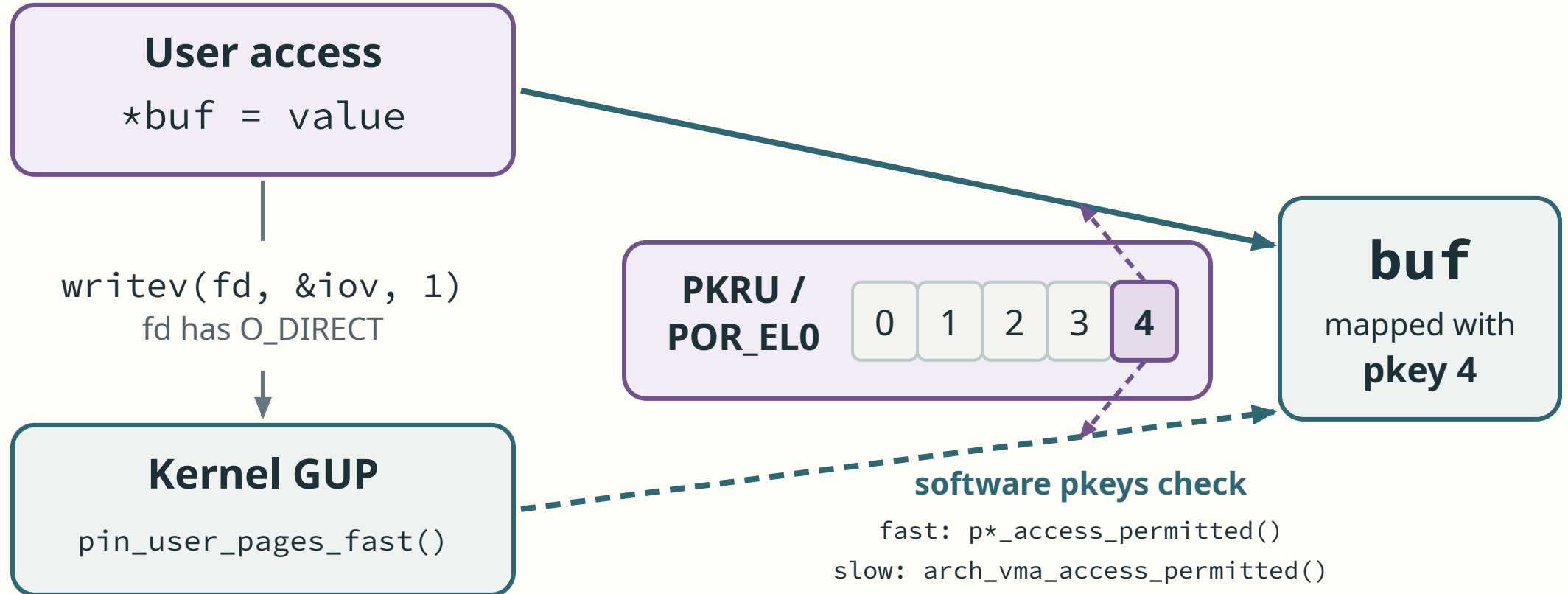
TL;DR: no consistent pkeys policy for **out-of-context uaccess/GUP**

# In-context uaccess



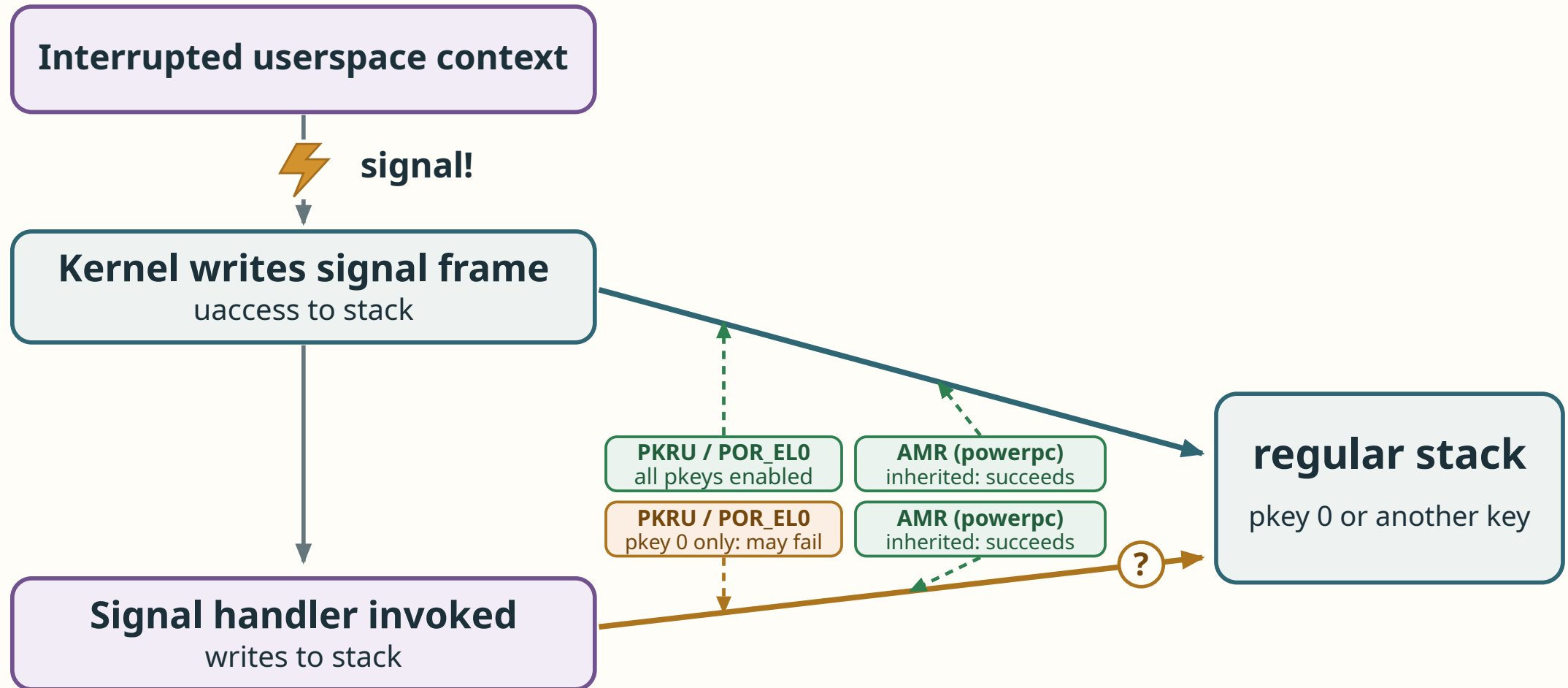
✓ pkeys permissions inherited from the caller

# In-context GUP

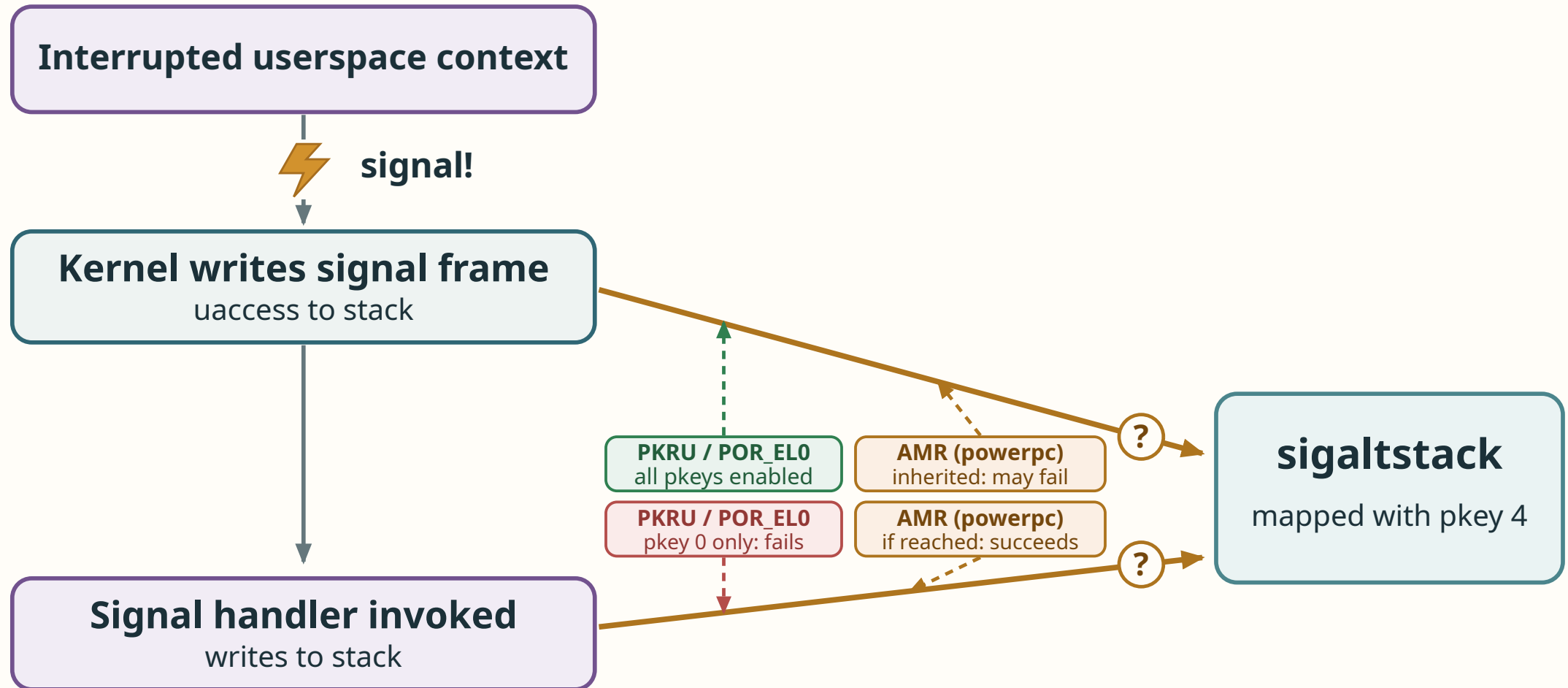


✓ pkeys permissions inherited from the caller

# Signal delivery: regular stack



# Signal delivery: sigaltstack (pkey 4)



# Out-of-context uaccess/GUP

## In-thread async uaccess

- **rseq:** kernel writes registered struct rseq on the return-to-user path
  - Proposed reset to pkey 0 only: [PATCH v7 0/4] rseq: Make rseq work with protection keys [2025]
- **CLONE\_CHILD\_CLEARTID:** on exit, write zero to the TID address supplied at clone; wake a futex waiter
- **Robust futexes:** on exit, walk the registered userspace list; mark owned locks dead and wake waiters
  - List reads and futex-word update: uaccess
  - Non-PI wake: shared-key lookup uses GUP

### Current policy

Inherit the current thread's pkeys state

## Worker threads

- **io\_uring:** queued work may access user memory via uaccess or GUP
- **vhost:** workers access virtqueue memory via uaccess; some paths use GUP

### Current policy

|         | Worker thread           | kthread                    |
|---------|-------------------------|----------------------------|
| x86     | Inherit creator's PKRU  | init PKRU (pkey 0 default) |
| arm64   | POR_EL0: pkey 0         | POR_EL0: pkey 0            |
| powerpc | AMR reset to permissive | Default AMR (user pkey 0)  |

**Worker thread:** PF\_USER\_WORKER, shared mm (io\_uring; default vhost).  
**kthread:** PF\_KTHREAD, borrows mm via kthread\_use\_mm() (vhost mode).

# pkeys bypass via remote GUP

- `process_vm_{readv,writev}(self)` bypass pkeys restrictions
- `/proc/self/mem` read/write do too
- Both are always permitted when targeting `self`
- Both use remote GUP, which skips the target pkeys check even for `self`

# Proposals discussed so far

## Signal policy

- `pkey_alloc(PKEY_ALLOC_SETSIGNAL, <rights>)`
  - Signal handlers also run with `<rights>` for this pkey
  - pkeys: Support setting access rights for signal handlers [2017]
- `PKEY_ALLOC_SIGNALINHERIT`
  - Always inherit the pkeys register (powerpc behaviour)
  - mm, x86: pkeys: Introduce `PKEY_ALLOC_SIGNALINHERIT` and change signal semantics [2018]

## Beyond signals

- Reserve a pkey for rseq/TLS that is always accessible [but: very few pkeys on arm64!]
- Use the perms defined by `PKEY_ALLOC_SETSIGNAL` for rseq and maybe other async uaccess cases
  - Question: **extend** the inherited pkeys reg, or **reset** to defaults first?

### Discussion threads

Re: [PATCH v4 00/14] RELRO link maps [2025]

Re: [PATCH v7 3/4] rseq: Make rseq work with protection keys [2025]

# Discussion: what policy? Defined how?

- In-thread async events: signal handlers, rseq, robust futexes, etc.
  - What should be user-configurable, and how?
  - Should we ever inherit the interrupted context?
- Worker threads: io\_uring, vhost, etc.
  - Step 1: align architectures!
    - pkey 0 everywhere?
  - Step 2: add configurability
- Does the remote GUP bypass matter?

pkeys are one example. **More features with similar mechanisms are coming on arm64.**

# arm

Merci

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Thank You

감사합니다

धन्यवाद

Kiitos

شكراً

ধন্যবাদ

ధన్యవాదములు

Köszönöm



# arm

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

[www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks)

