Linux Kernel Network Security - Transport Layer Security (TLS)

Deep Hacking

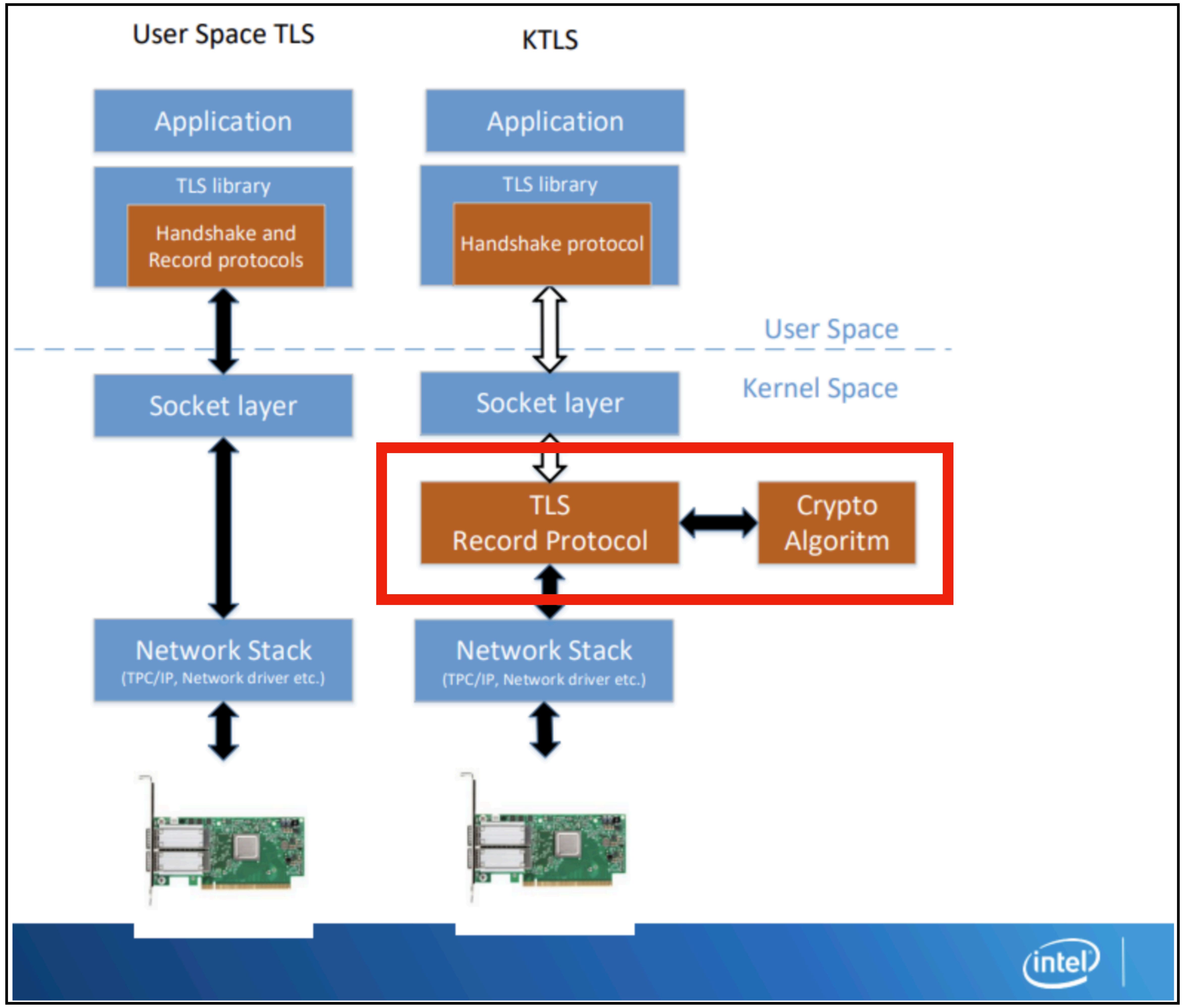
2025/01/19 Pumpkin 🎃

Outline

- Overview
- Vulnerability

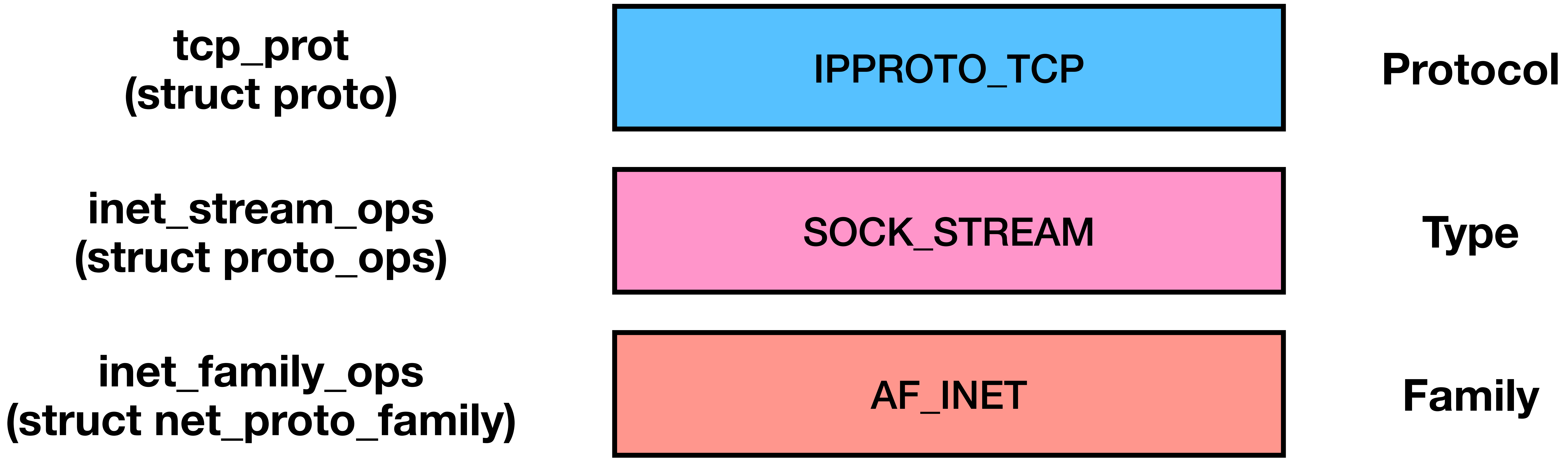
Overview

Overview

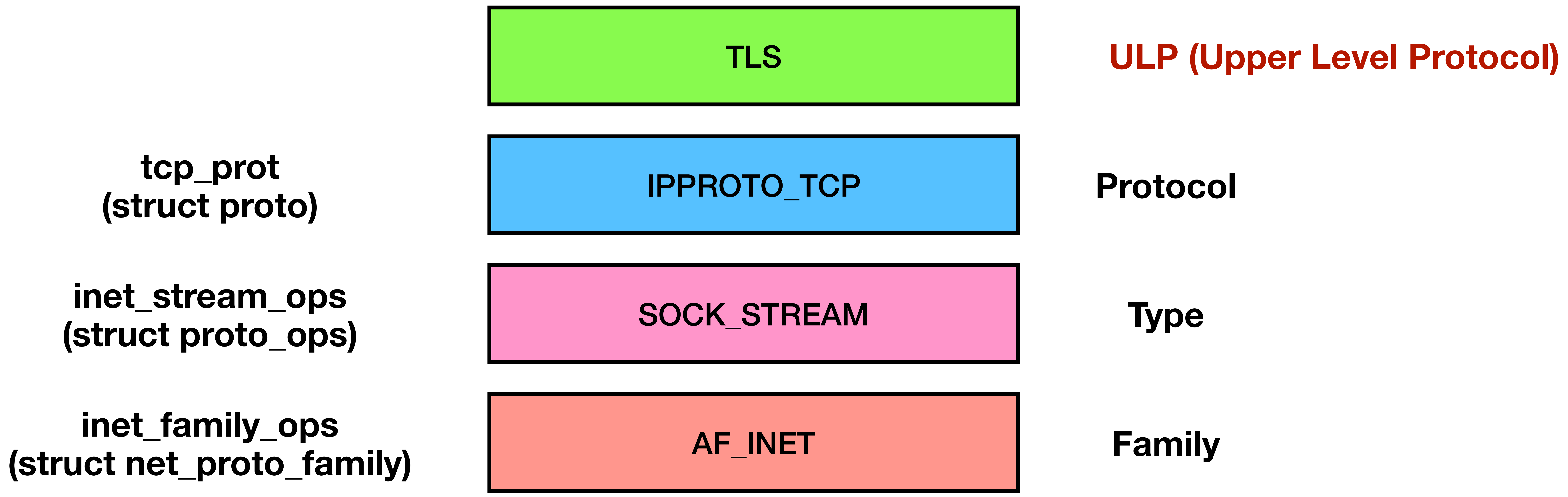


https://blog.salrashid.dev/articles/2022/kernel_tls/

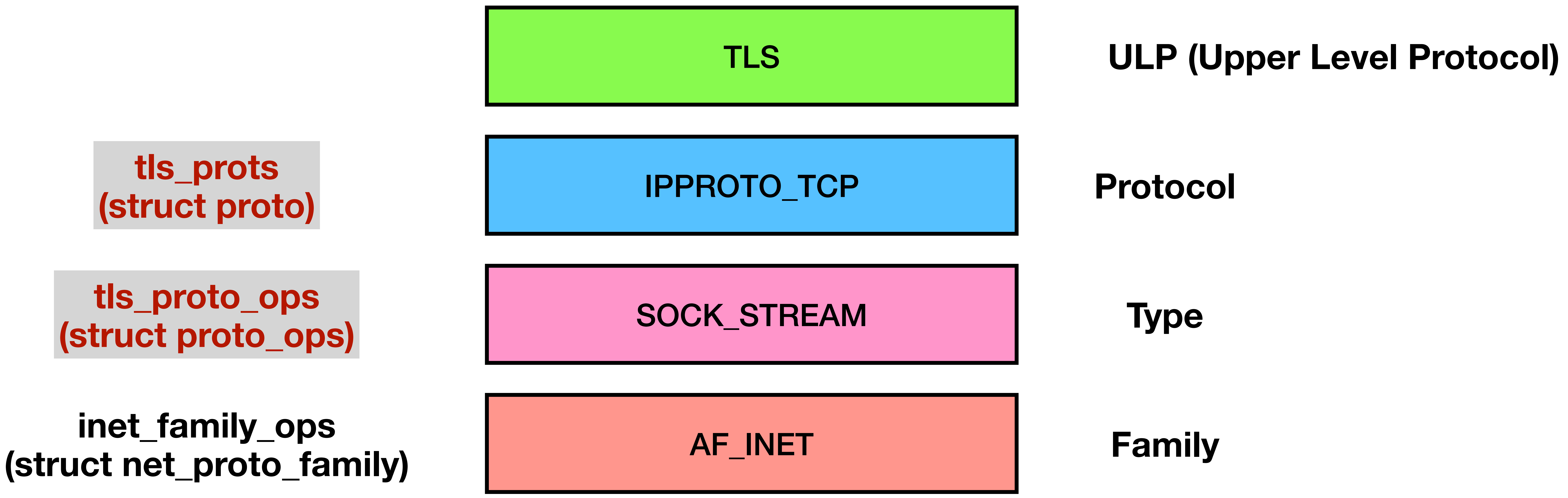
Overview

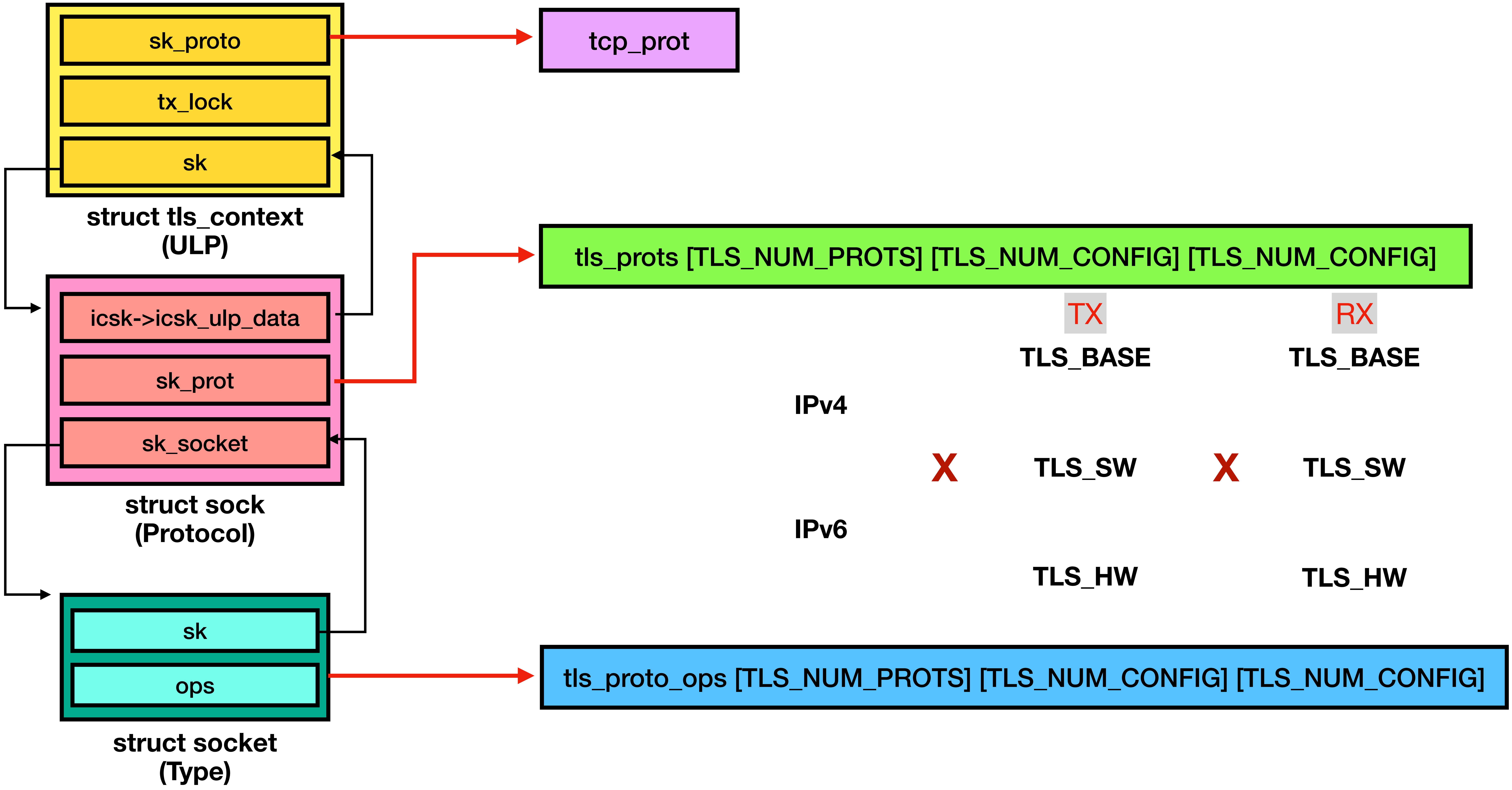

tcp_prot
(struct proto)
IPPROTO_TCP
Protocol
inet_stream_ops
(struct proto_ops)
SOCK_STREAM
Type
inet_family_ops
(struct net_proto_family)
AF_INET
Family

Overview

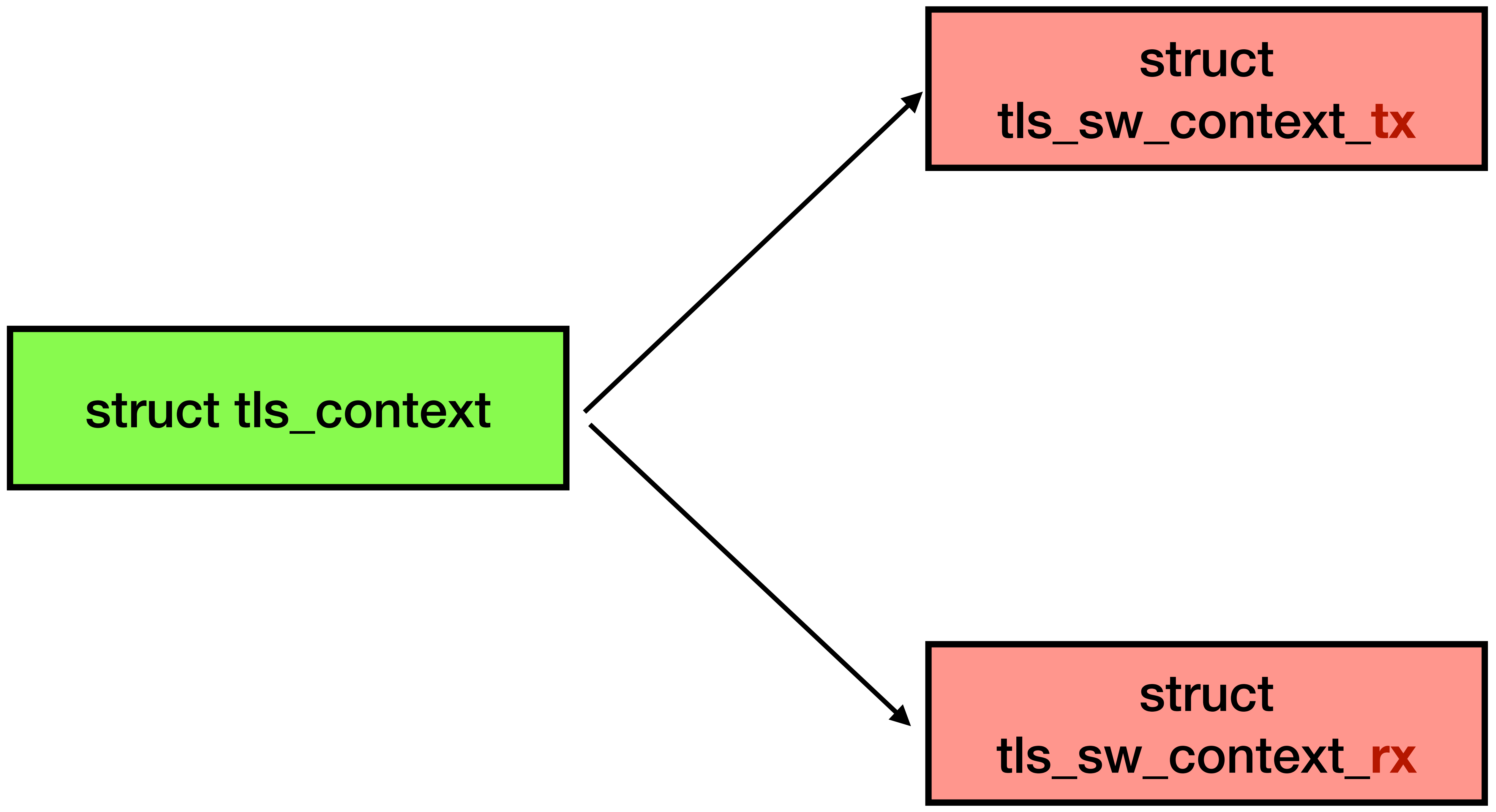

TLS
ULP (Upper Level Protocol)
tcp_prot
(struct proto)
IPPROTO_TCP
Protocol
inet_stream_ops
(struct proto_ops)
SOCK_STREAM
Type
inet_family_ops
(struct net_proto_family)
AF_INET
Family

Overview


TLS
ULP (Upper Level Protocol)
tls_prots
(struct proto)
IPPROTO_TCP
Protocol
tls_proto_ops
(struct proto_ops)
SOCK_STREAM
Type
inet_family_ops
(struct net_proto_family)
AF_INET
Family


sk_proto
tx_lock
sk
struct tls_context (ULP)
tcp_prot
icsk->icsk_ulp_data
sk_prot
sk_socket
struct sock (Protocol)
tls_prots [TLS_NUM_PROTS] [TLS_NUM_CONFIG] [TLS_NUM_CONFIG]
TX
RX
TLS_BASE
TLS_BASE
IPv4
X
TLS_SW
X
TLS_SW
IPv6
TLS_HW
TLS_HW
sk
ops
struct socket (Type)
tls_proto_ops [TLS_NUM_PROTS] [TLS_NUM_CONFIG] [TLS_NUM_CONFIG]

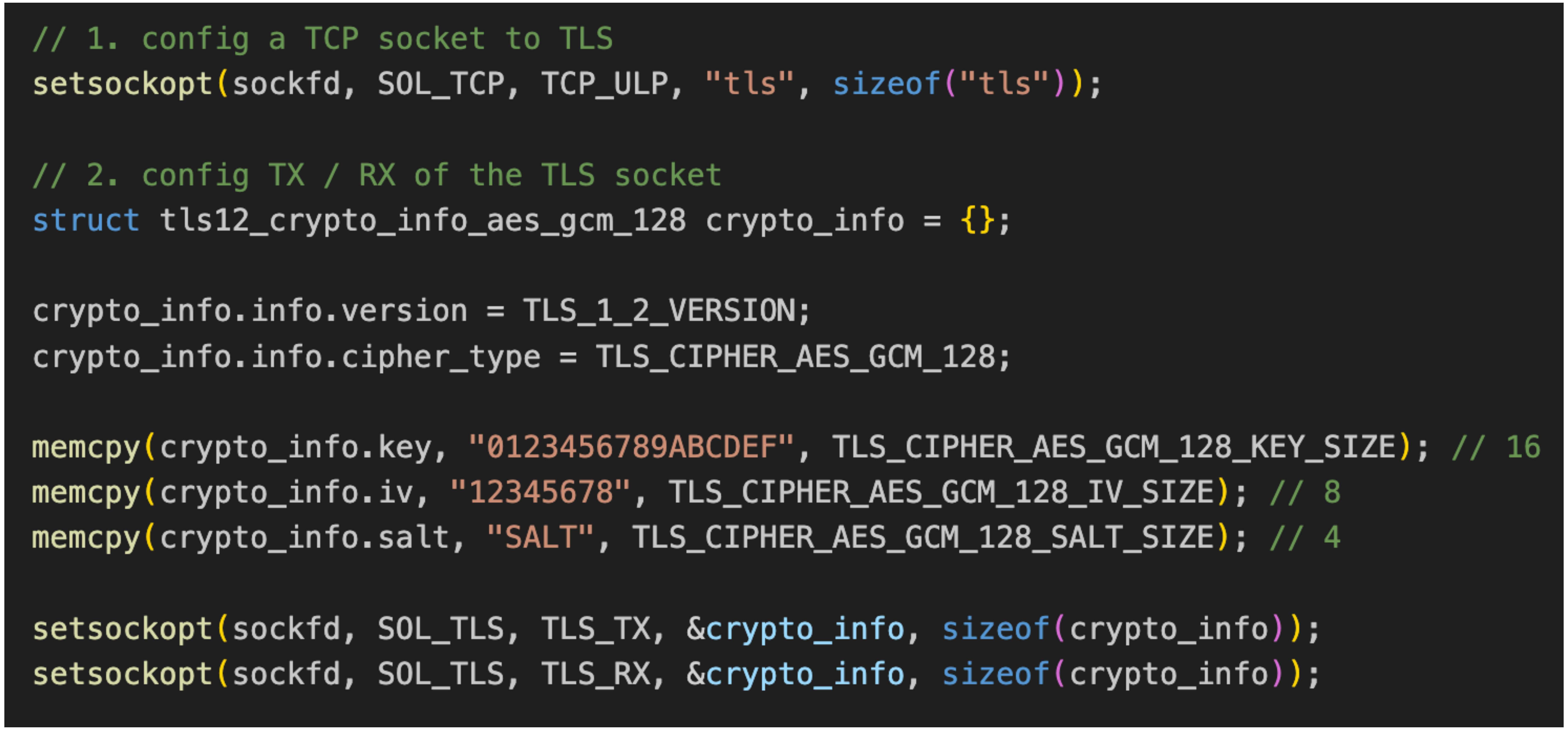
Overview



cipher type	AES_GCM_128
TLS vers.	TLS_1_2
KEY	0123…DEF
IV	12345678
SALT	SALT
…	…

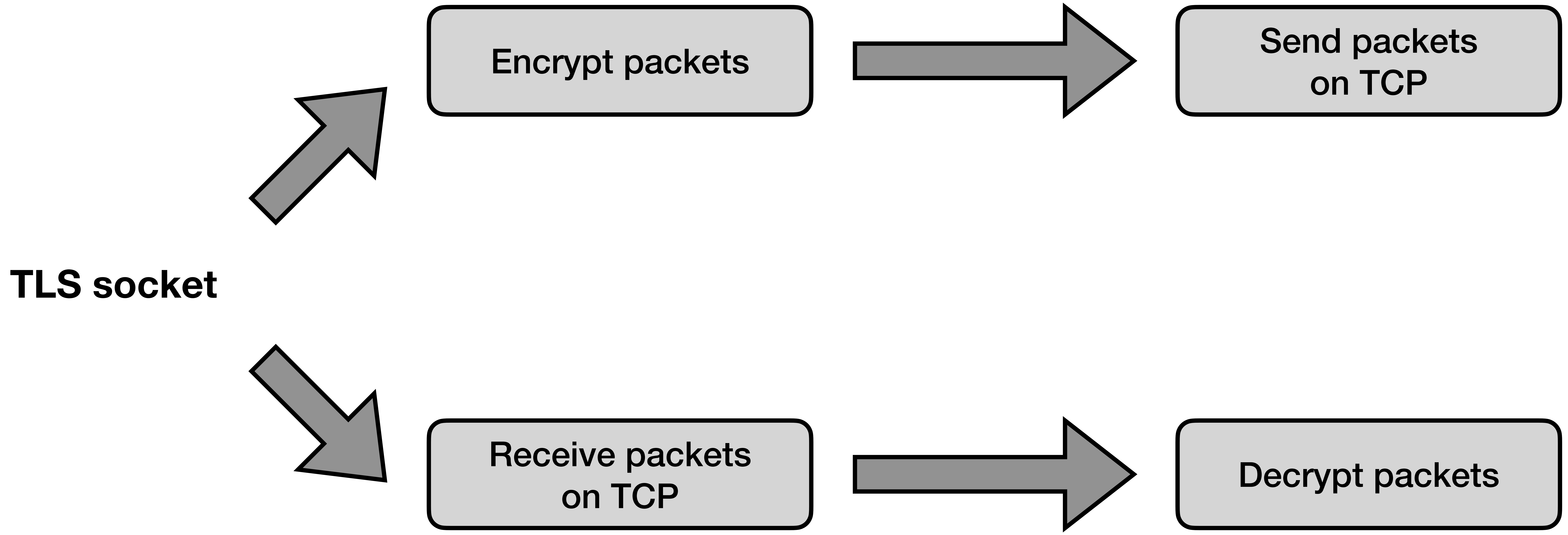
cipher type	…
TLS vers.	…
KEY	…
IV	…
SALT	…
…	…

Overview

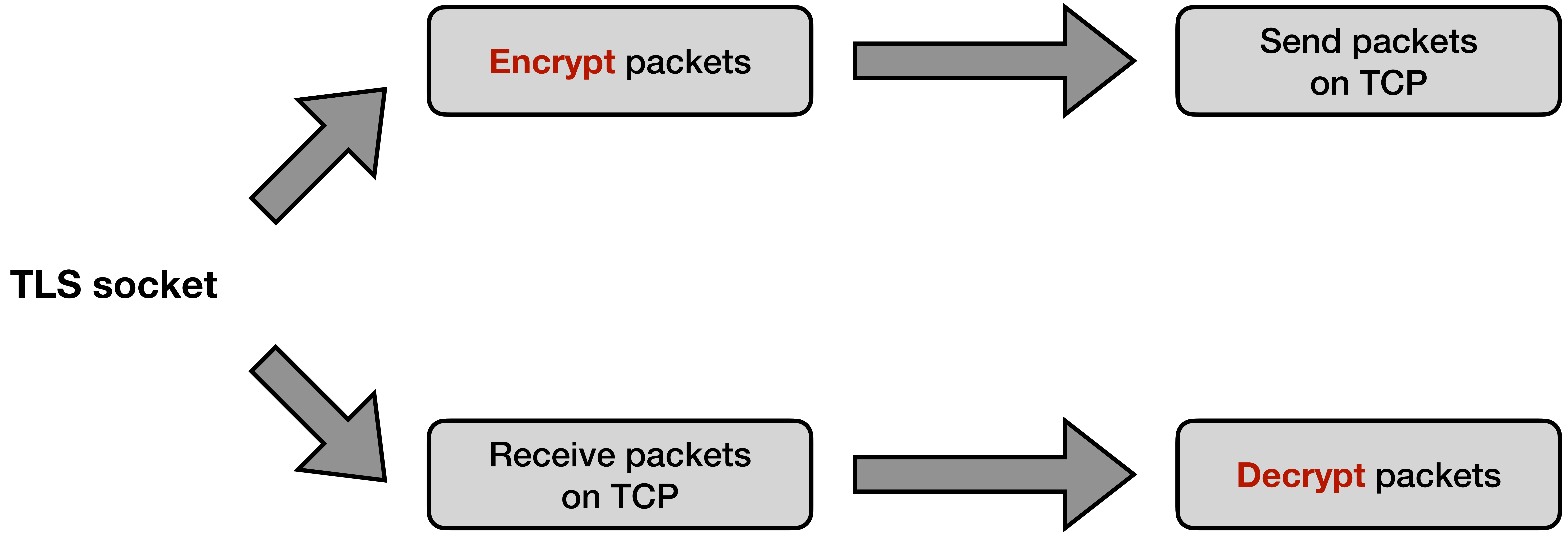


```c
// 1. config a TCP socket to TLS
setsockopt(sockfd, SOL_TCP, TCP_ULP, "tls", sizeof("tls"));

// 2. config TX / RX of the TLS socket
struct tls12_crypto_info_aes_gcm_128 crypto_info = {};

crypto_info.info.version = TLS_1_2_VERSION;
crypto_info.info.cipher_type = TLS_CIPHER_AES_GCM_128;

memcpy(crypto_info.key, "0123456789ABCDEF", TLS_CIPHER_AES_GCM_128_KEY_SIZE); // 16
memcpy(crypto_info.iv, "12345678", TLS_CIPHER_AES_GCM_128_IV_SIZE); // 8
memcpy(crypto_info.salt, "SALT", TLS_CIPHER_AES_GCM_128_SALT_SIZE); // 4

setsockopt(sockfd, SOL_TLS, TLS_TX, &crypto_info, sizeof(crypto_info));
setsockopt(sockfd, SOL_TLS, TLS_RX, &crypto_info, sizeof(crypto_info));
```

Overview


TLS socket
Encrypt packets
Send packets on TCP
Receive packets on TCP
Decrypt packets

Overview


TLS socket
Encrypt packets
Send packets on TCP
Receive packets on TCP
Decrypt packets

Overview

- Supported TLS algorithms
    - gcm(aes)
    - ccm(aes)
    - gcm(sm4)
    - …

Overview

- Supported TLS algorithms
    - **gcm**(aes)
    - **ccm**(aes)
    - **gcm**(sm4)
    - …   **Template name**

Overview

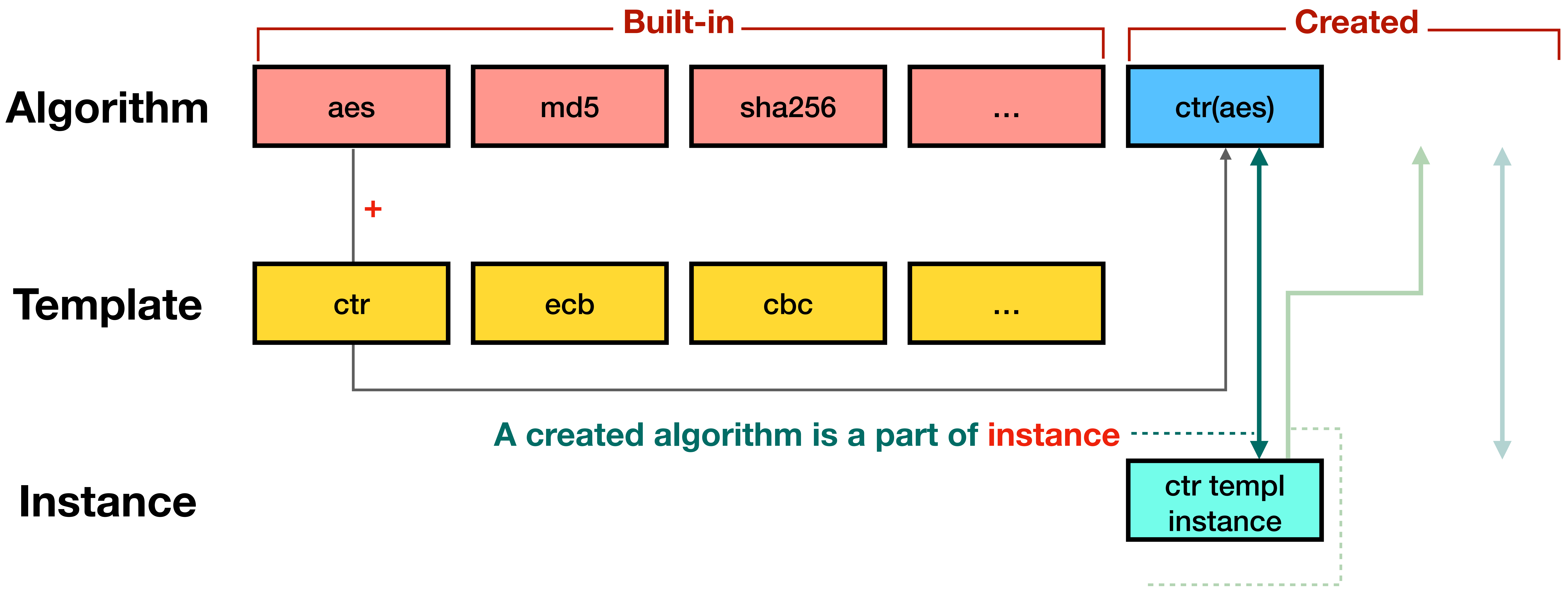
- Supported TLS algorithms
  - gcm(**aes**)
  - ccm(**aes**)
  - gcm(**sm4**)
  - … **Cipher name**

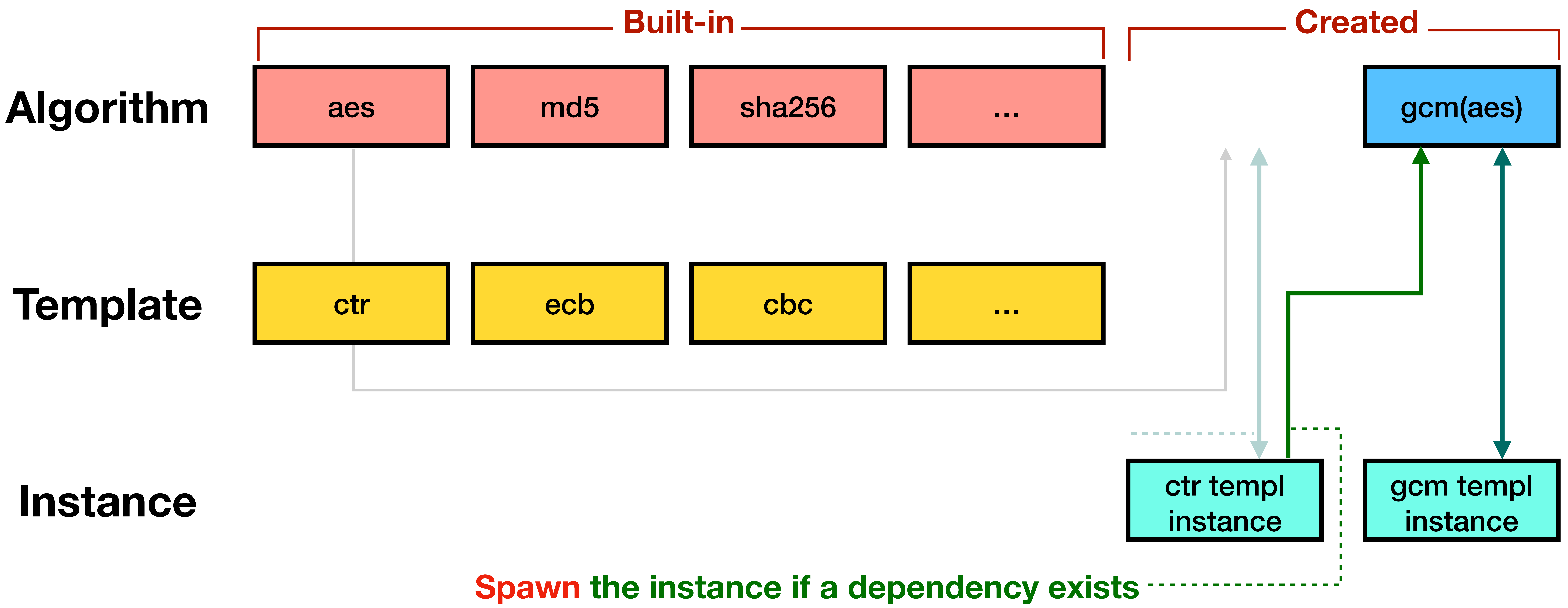
Overview

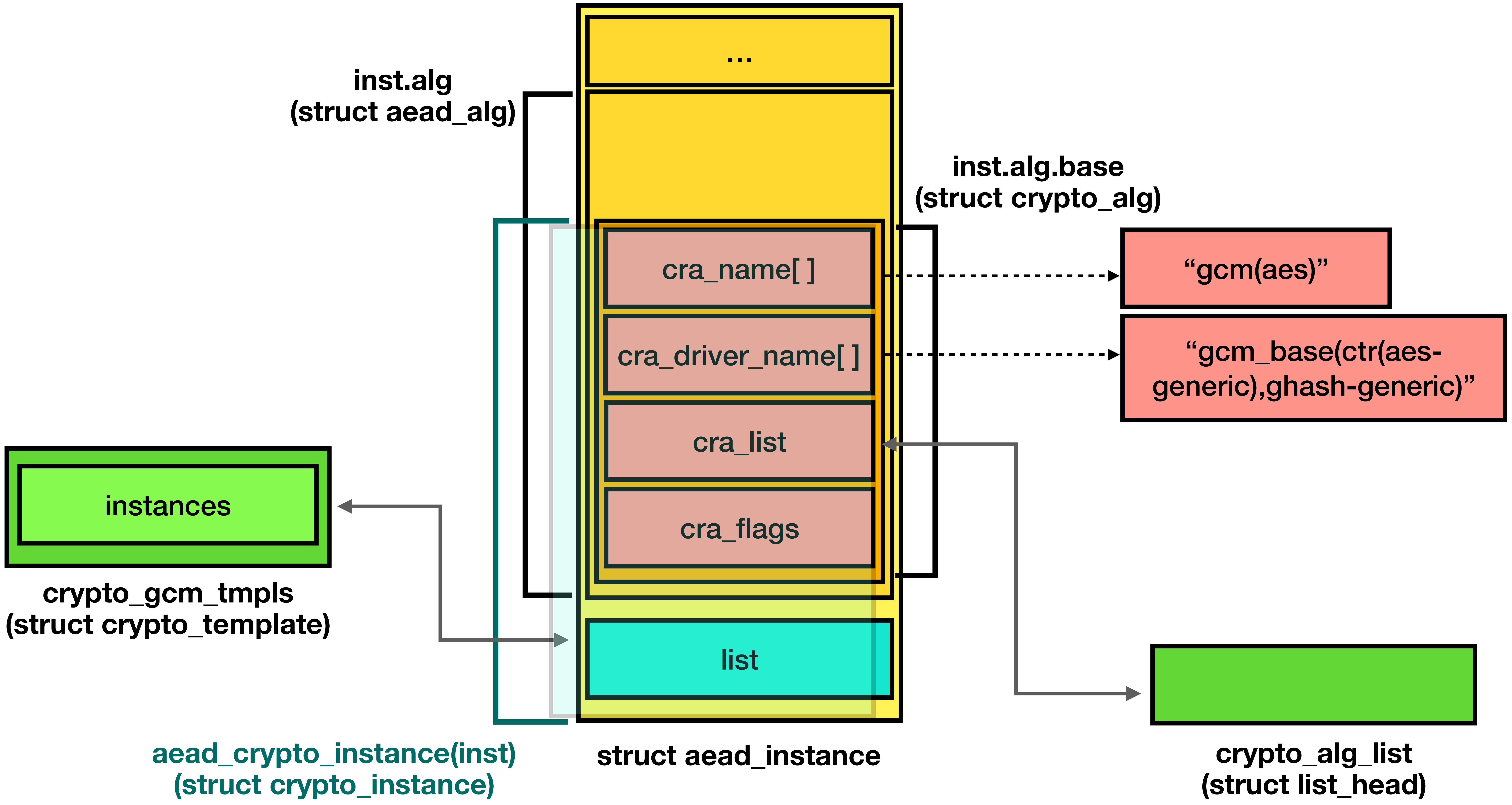
- **Algorithm**
  - Implementation of a specific cryptographic operation, such as AES, SHA-256, or HMAC
- **Template**
  - Constructing more complex cryptographic transformations by combining or layering simpler algorithms

Overview

- **Instance**
  - Instantiation of a cryptographic template, where specific algorithms and parameters have been configured
- **Spawn**
  - Create a linkage or dependency between cryptographic instances and algorithms


Built-in
Created
Algorithm
aes
md5
sha256
...
ctr(aes)
+
Template
ctr
ecb
cbc
...
A created algorithm is a part of instance
Instance
ctr templ instance


Built-in
Created
Algorithm
aes
md5
sha256
...
gcm(aes)
Template
ctr
ecb
cbc
...
Instance
ctr templ instance
gcm templ instance
Spawn the instance if a dependency exists

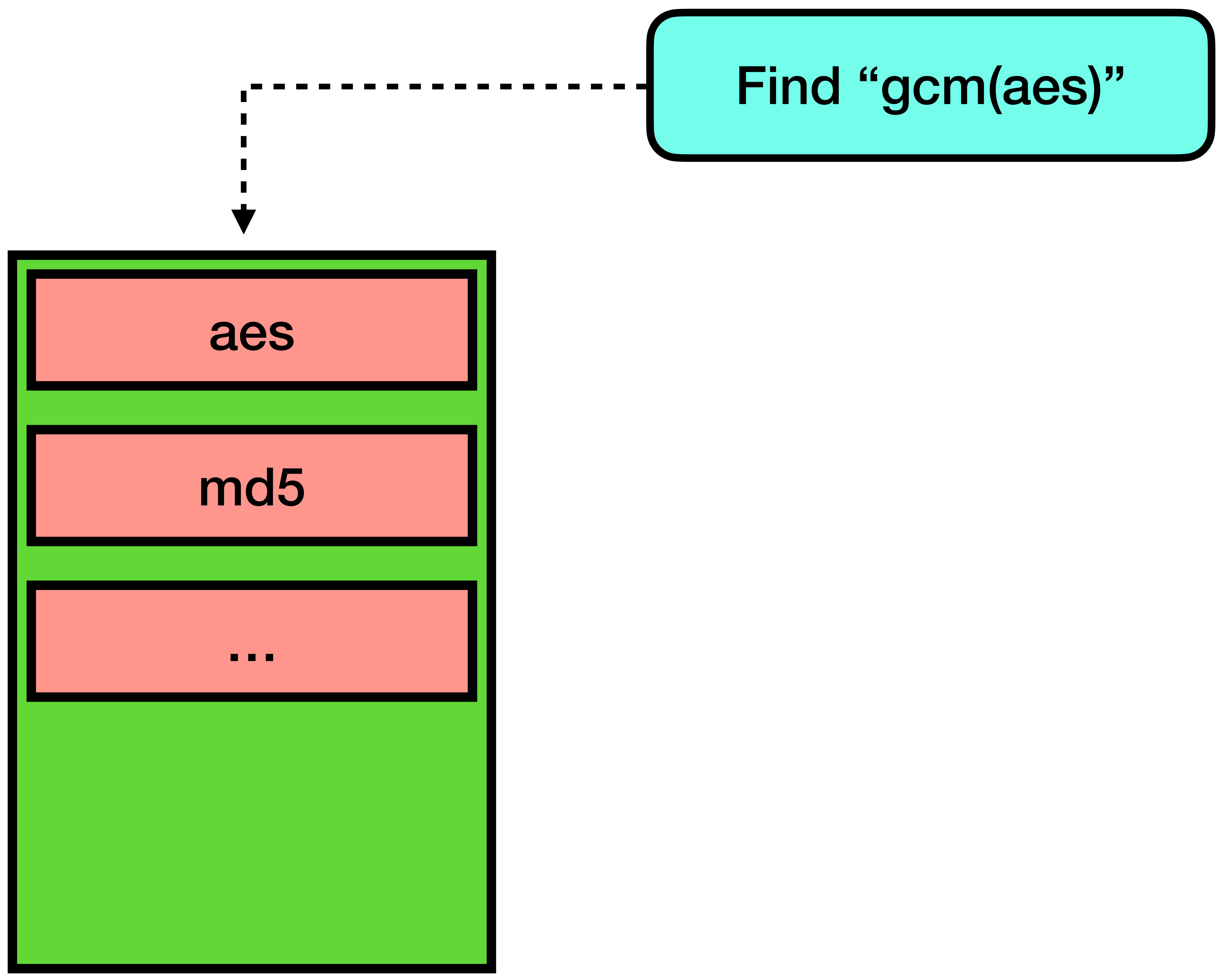

...
inst.alg
(struct aead_alg)
inst.alg.base
(struct crypto_alg)
cra_name[ ]
cra_driver_name[ ]
cra_list
cra_flags
"gcm(aes)"
"gcm_base(ctr(aes-generic),ghash-generic)"
instances
crypto_gcm_tmpls
(struct crypto_template)
list
aead_crypto_instance(inst)
(struct crypto_instance)
struct aead_instance
crypto_alg_list
(struct list_head)

Overview

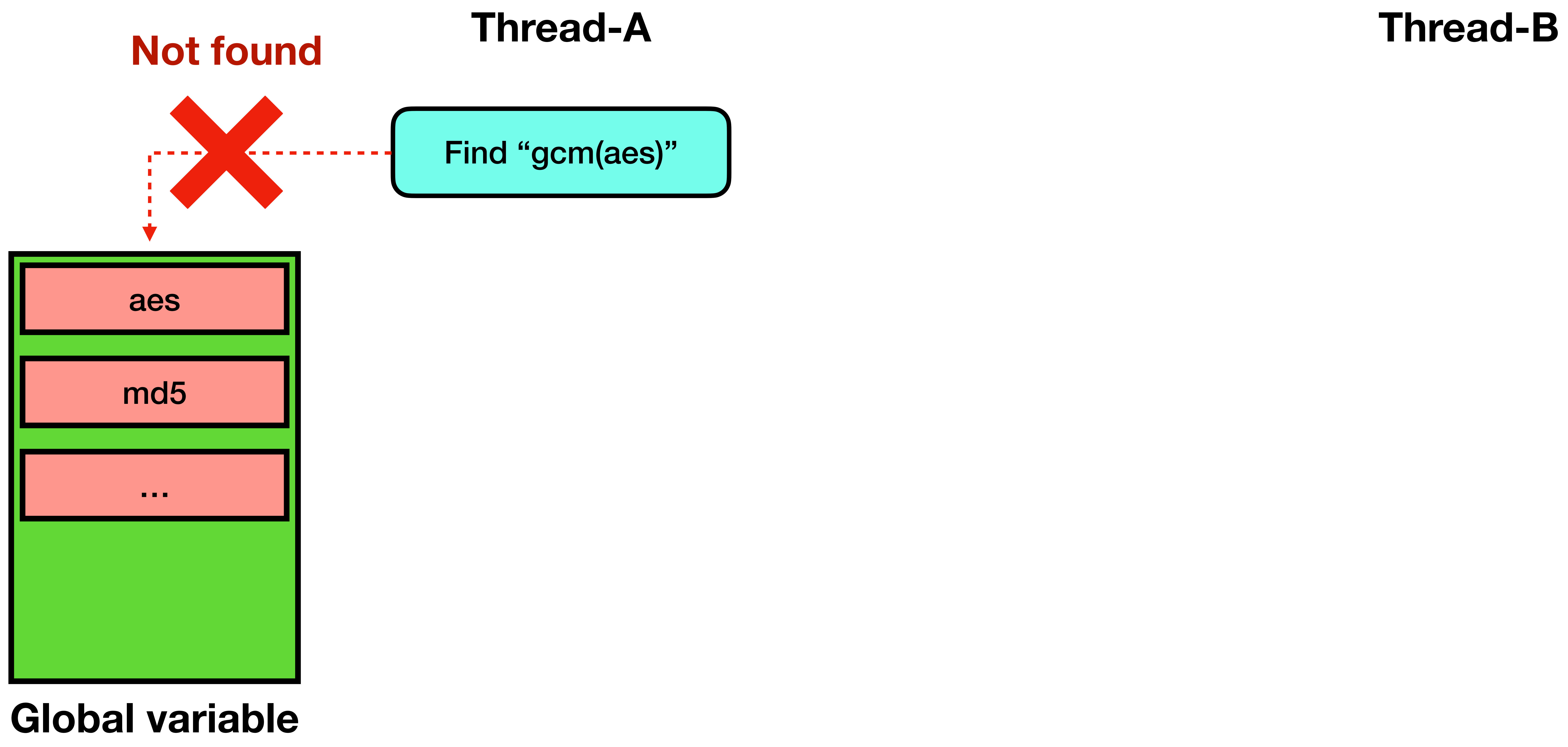
- For example, if we configure “gcm(aes)” as the crypto algorithm of TX…

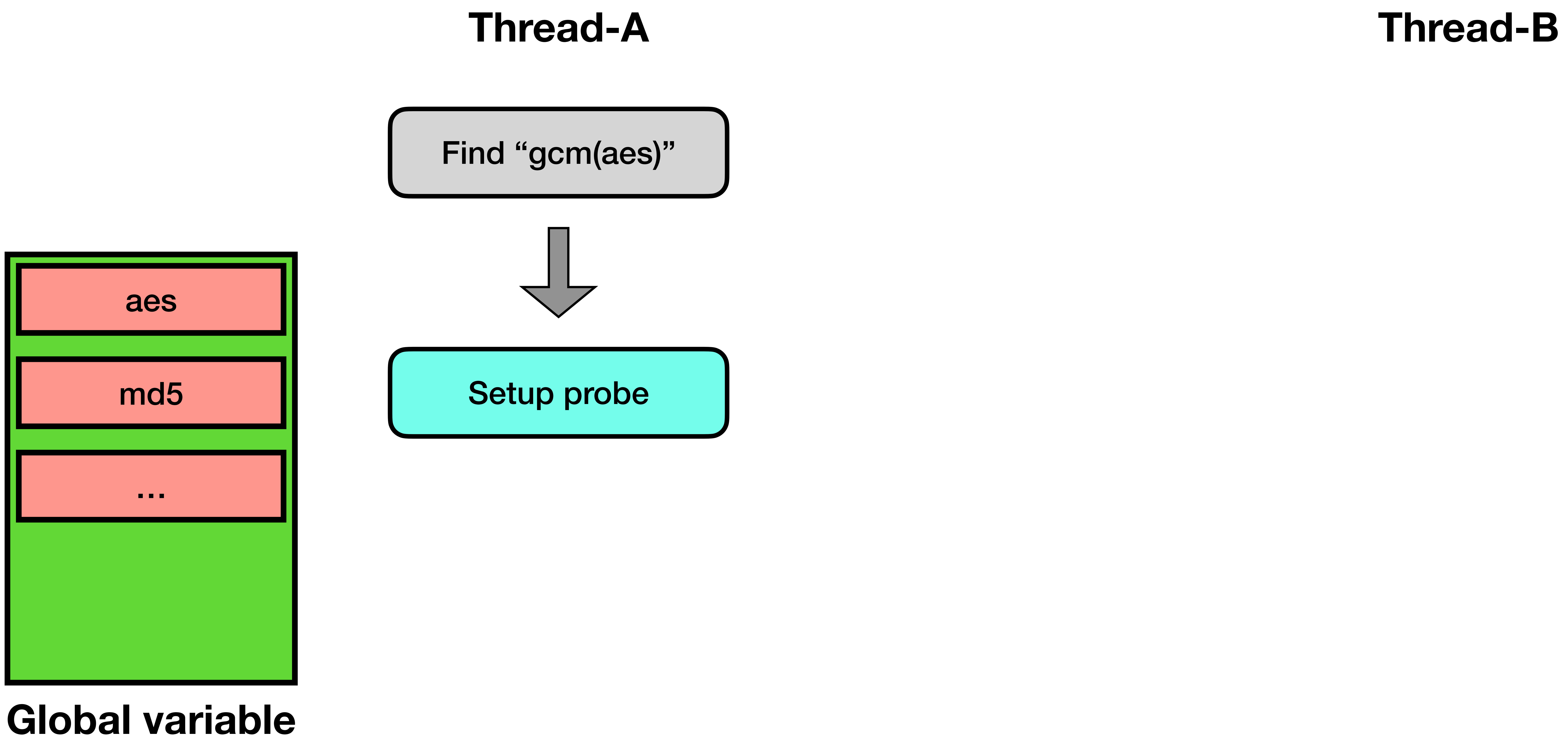
Thread-A

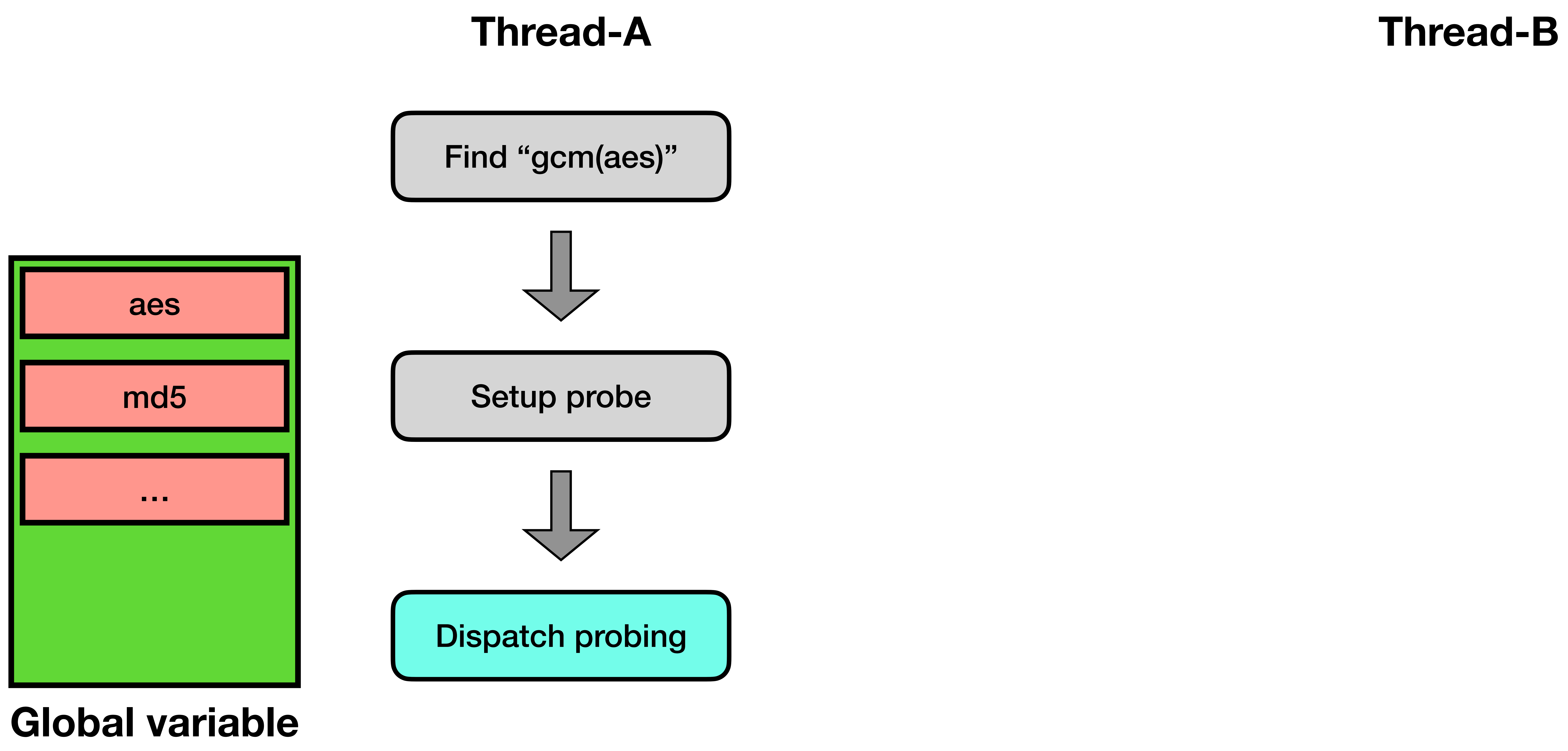
Thread-B

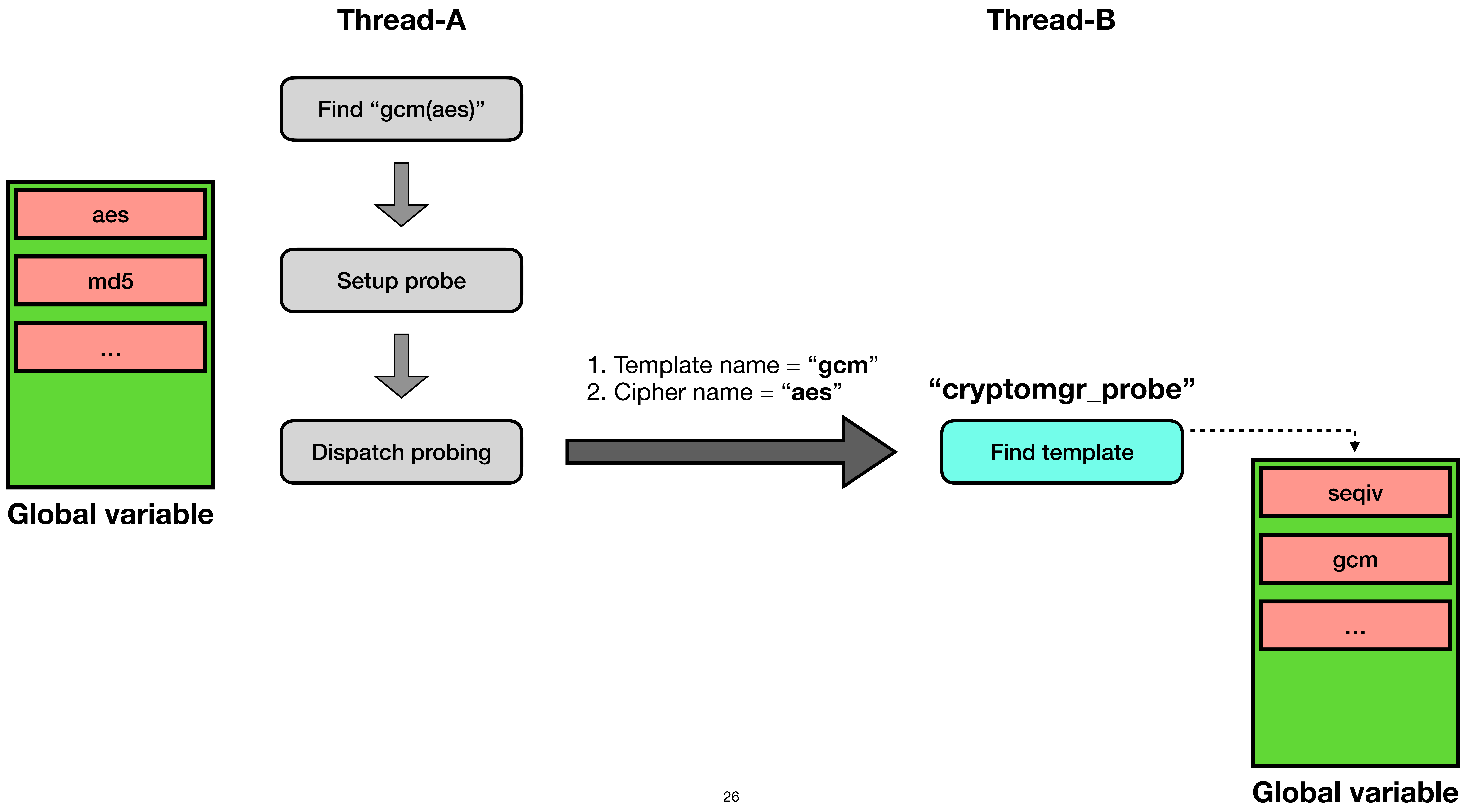


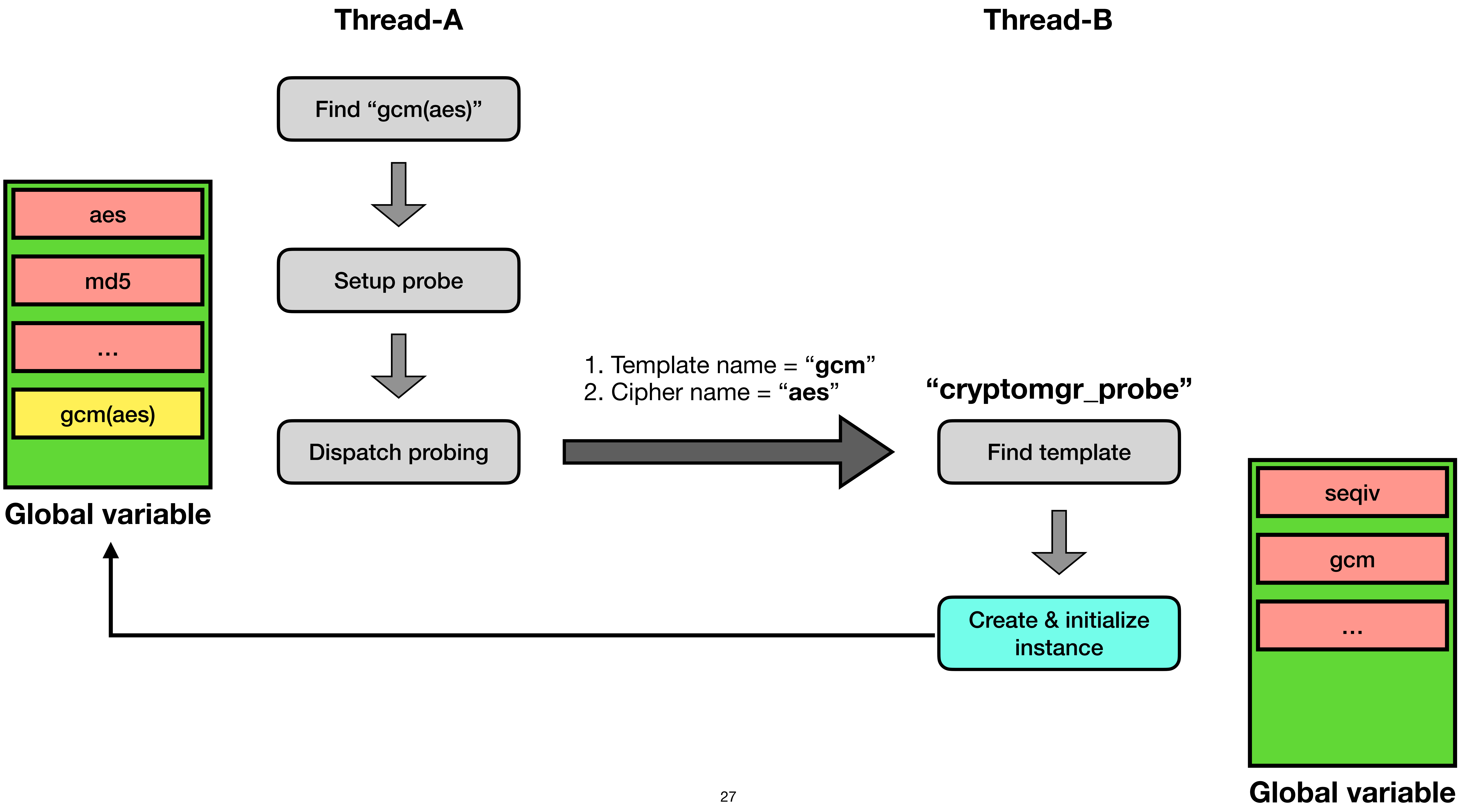
Global variable


Not found
Thread-A
Thread-B
Find "gcm(aes)"
aes
md5
...
Global variable


Thread-A
Thread-B
Find “gcm(aes)”
Setup probe
aes
md5
...
Global variable

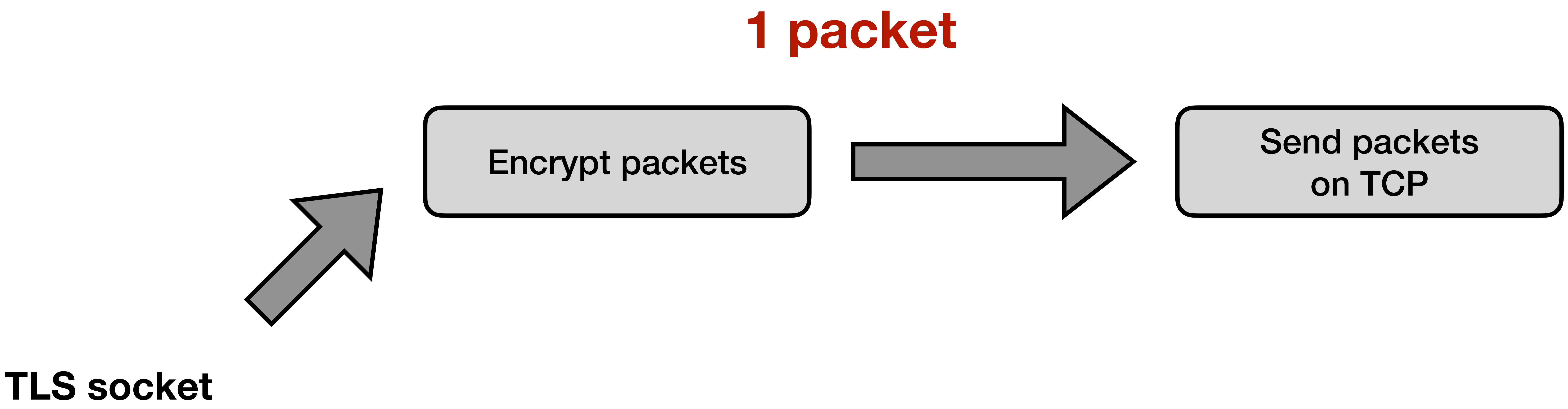

Thread-A
Thread-B
Find "gcm(aes)"
Setup probe
Dispatch probing
aes
md5
...
Global variable


Thread-A
Thread-B
Find “gcm(aes)”
Setup probe
Dispatch probing
aes
md5
...
Global variable
1. Template name = “**gcm**”
2. Cipher name = “**aes**”
“cryptomgr_probe”
Find template
seqiv
gcm
...
Global variable

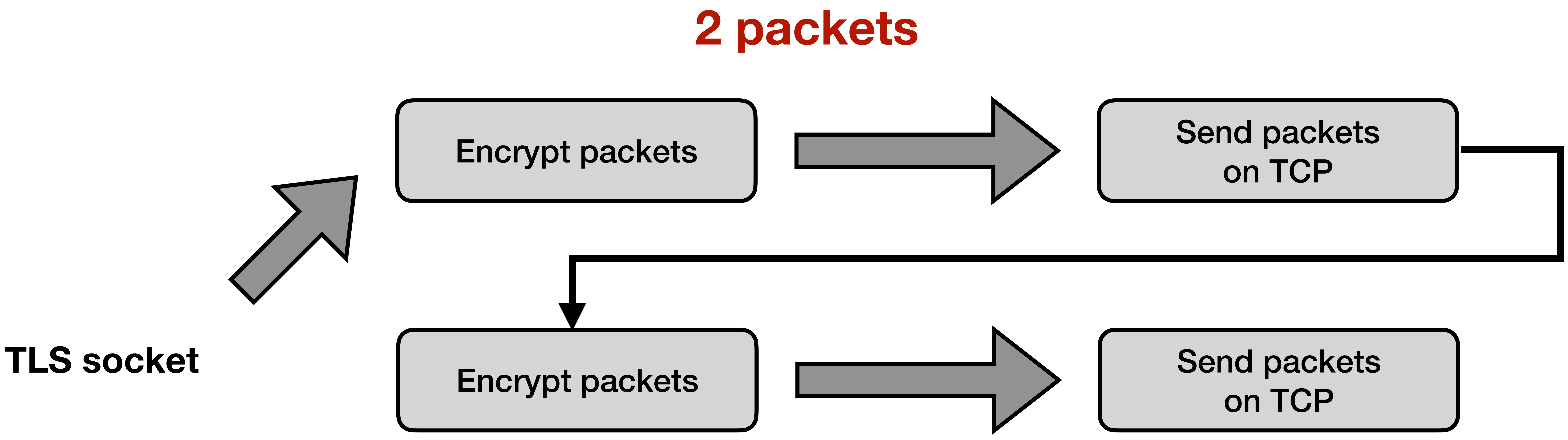


Thread-A
Thread-B
Find "gcm(aes)"
Setup probe
Dispatch probing
1. Template name = "**gcm**"
2. Cipher name = "**aes**"
"cryptomgr_probe"
Find template
Create & initialize instance
aes
md5
...
gcm(aes)
Global variable
seqiv
gcm
...
Global variable

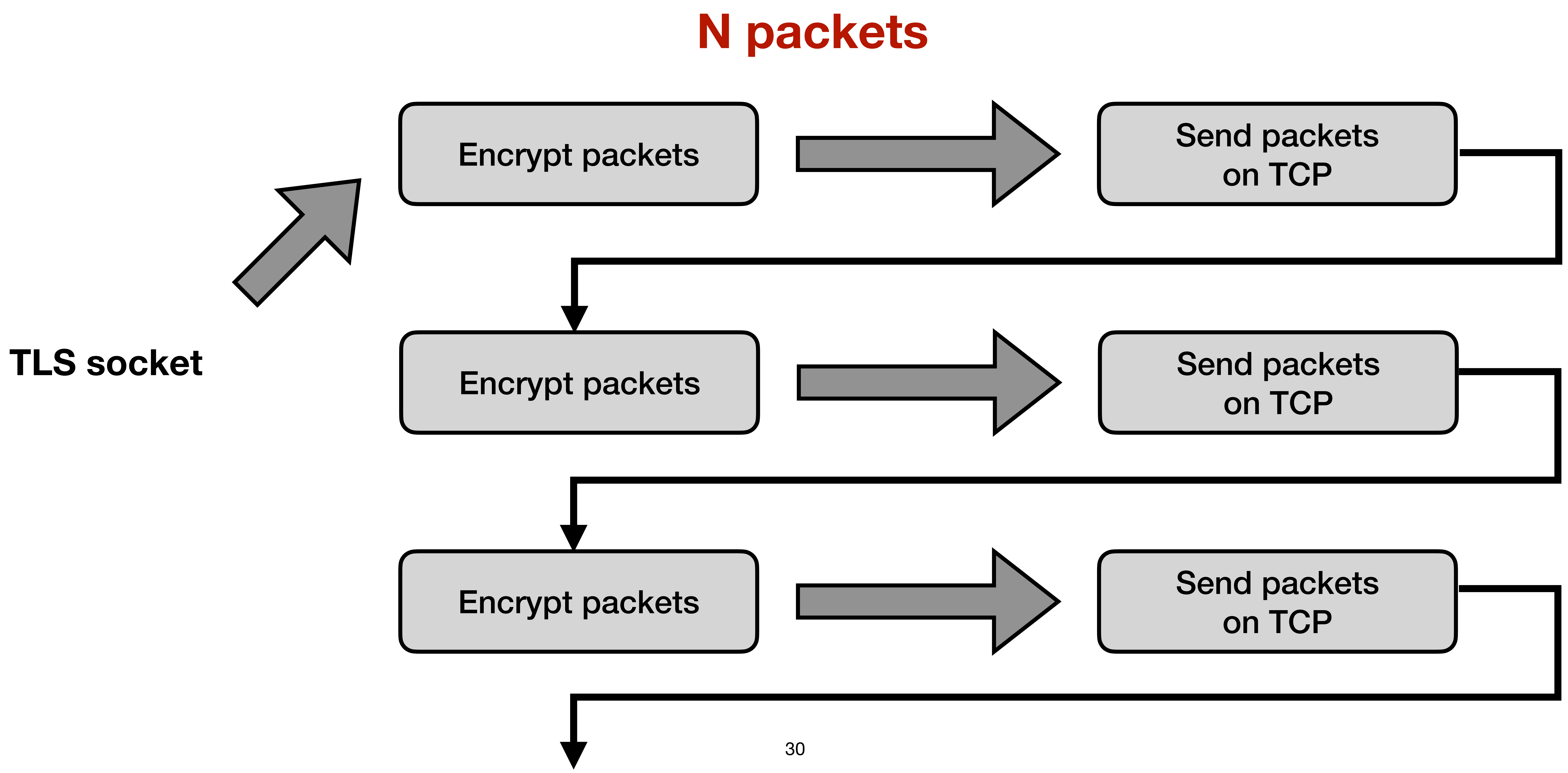
Overview



Overview

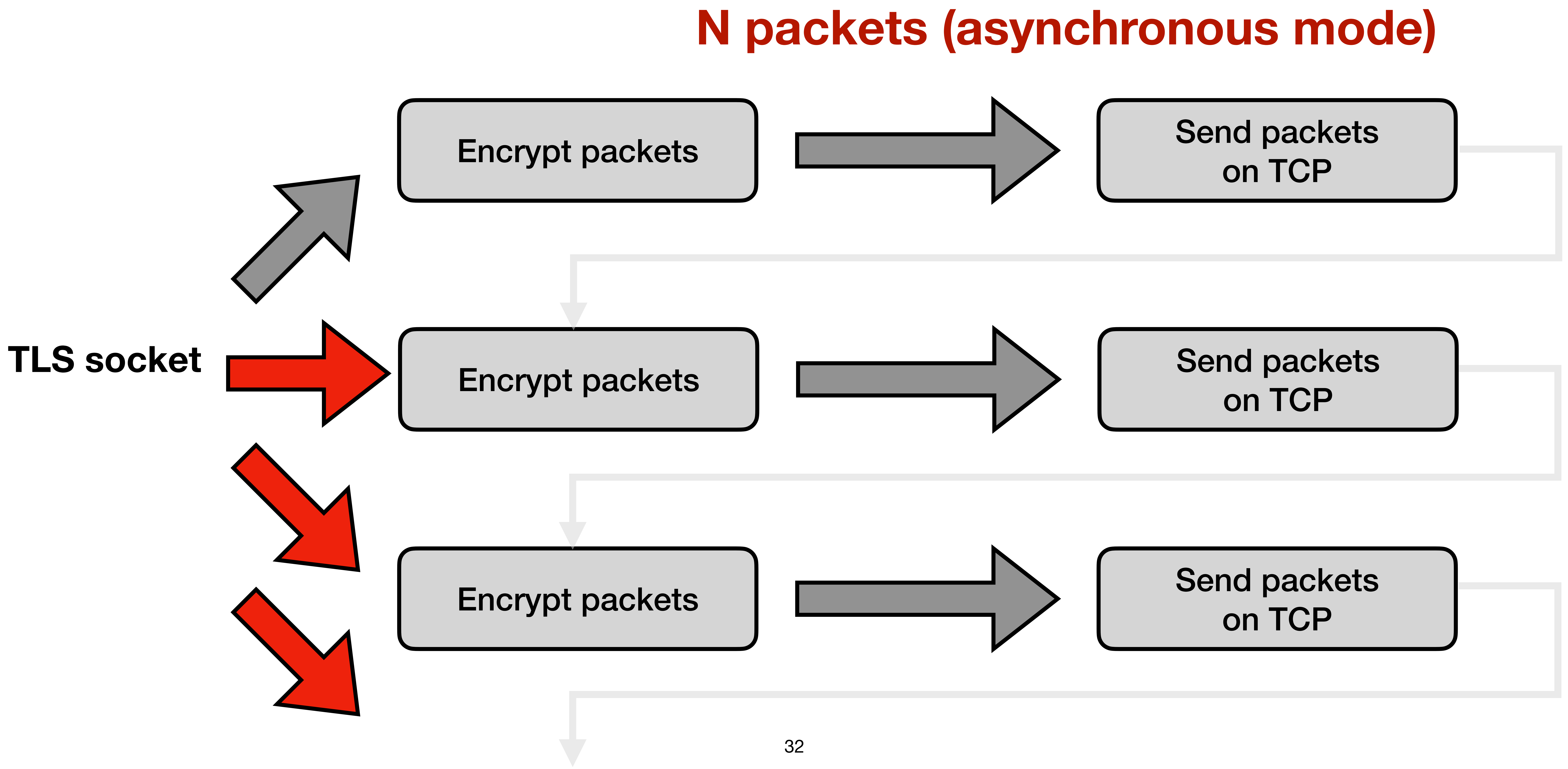

2 packets
Encrypt packets
Send packets on TCP
TLS socket
Encrypt packets
Send packets on TCP

Overview


N packets
TLS socket
Encrypt packets
Send packets on TCP
Encrypt packets
Send packets on TCP
Encrypt packets
Send packets on TCP

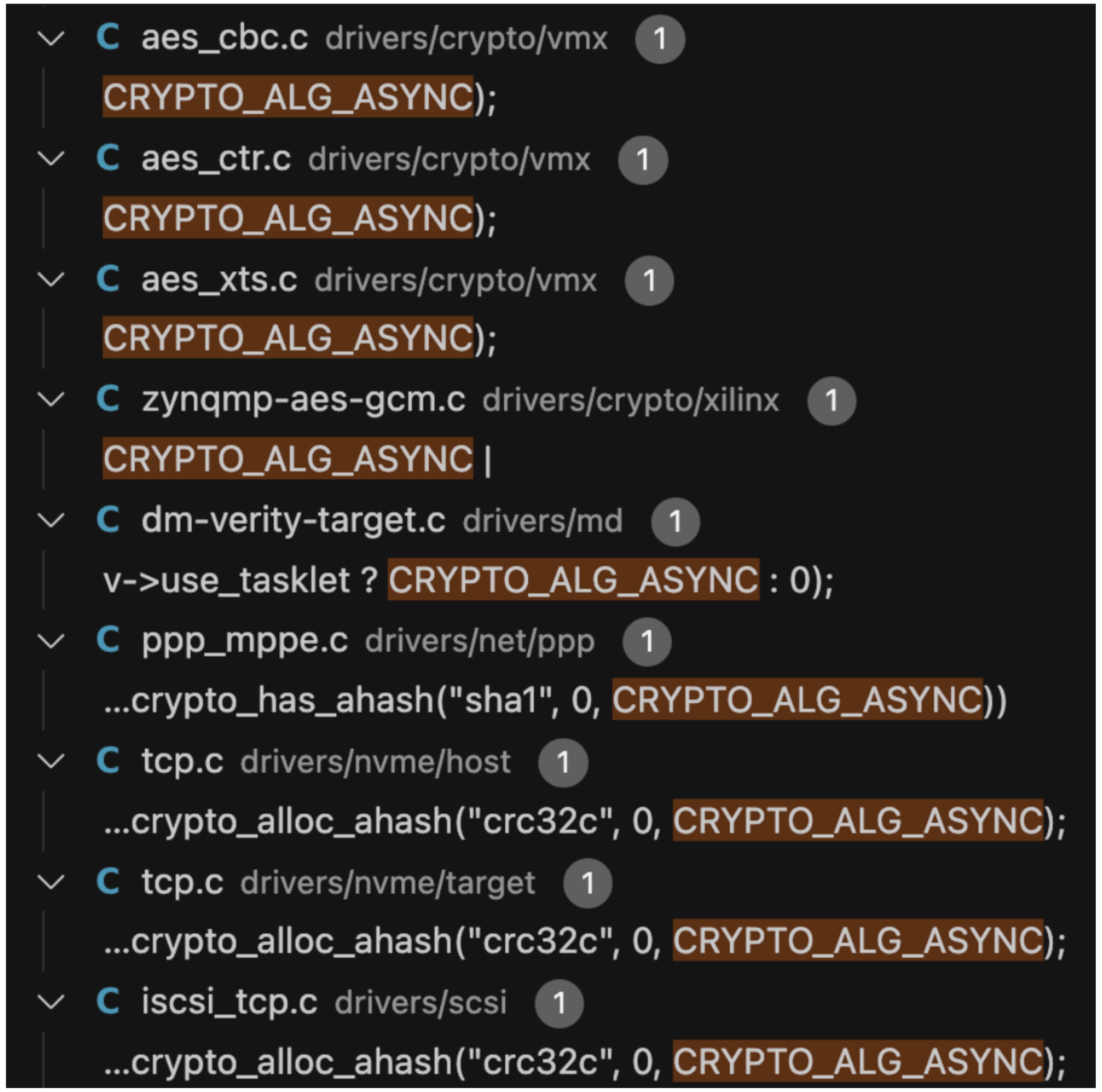
Overview

Overview


N packets (asynchronous mode)
TLS socket
Encrypt packets
Send packets on TCP
Encrypt packets
Send packets on TCP
Encrypt packets
Send packets on TCP

Overview

Vendor specific drivers



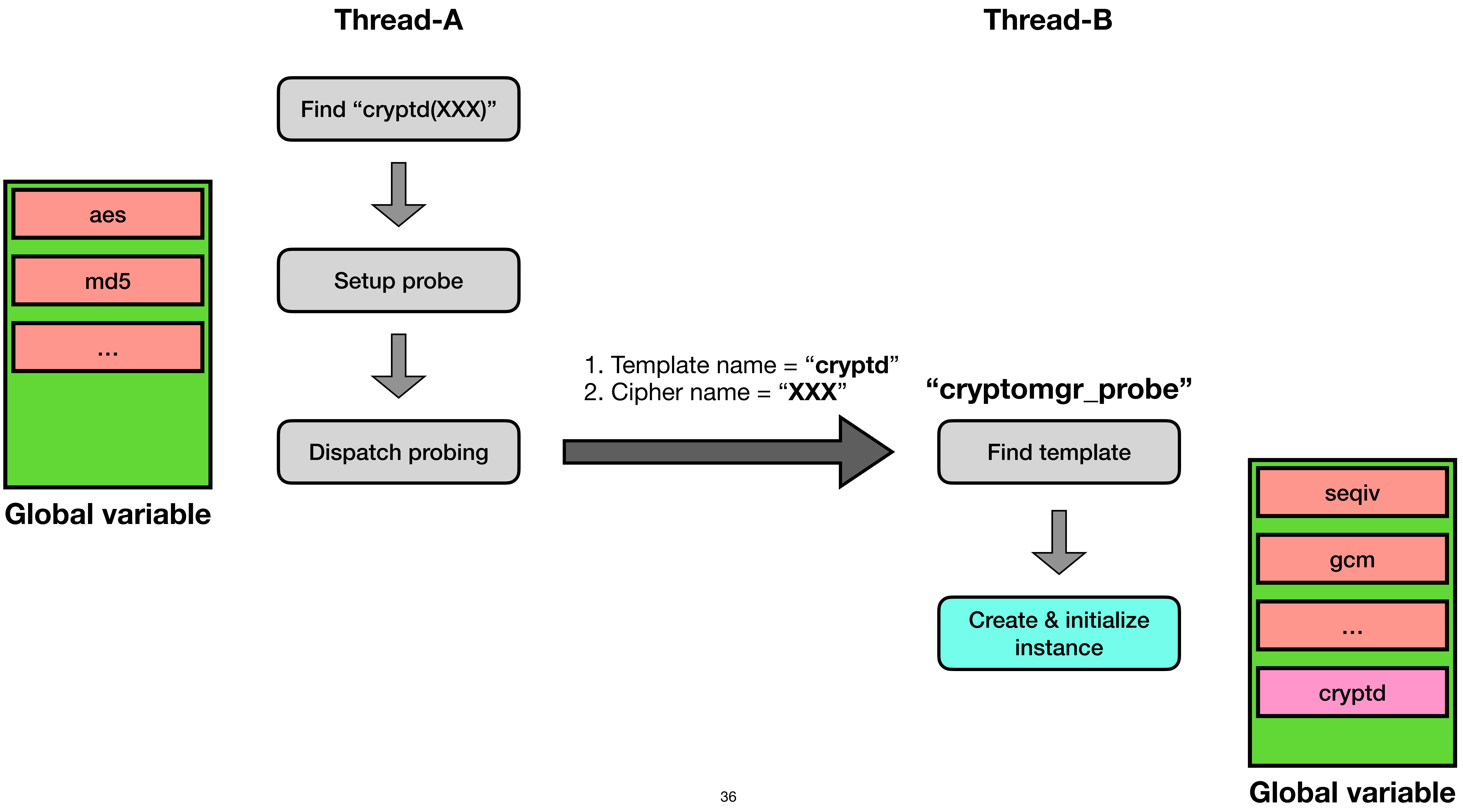
```
C aes_cbc.c drivers/crypto/vmx 1
  CRYPTO_ALG_ASYNC);
C aes_ctr.c drivers/crypto/vmx 1
  CRYPTO_ALG_ASYNC);
C aes_xts.c drivers/crypto/vmx 1
  CRYPTO_ALG_ASYNC);
C zynqmp-aes-gcm.c drivers/crypto/xilinx 1
  CRYPTO_ALG_ASYNC |
C dm-verity-target.c drivers/md 1
  v->use_tasklet ? CRYPTO_ALG_ASYNC : 0);
C ppp_mppe.c drivers/net/ppp 1
  ...crypto_has_ahash("sha1", 0, CRYPTO_ALG_ASYNC))
C tcp.c drivers/nvme/host 1
  ...crypto_alloc_ahash("crc32c", 0, CRYPTO_ALG_ASYNC);
C tcp.c drivers/nvme/target 1
  ...crypto_alloc_ahash("crc32c", 0, CRYPTO_ALG_ASYNC);
C iscsi_tcp.c drivers/scsi 1
  ...crypto_alloc_ahash("crc32c", 0, CRYPTO_ALG_ASYNC);
```

Overview

- **Cryptd**
  - Enabled when the CONFIG_CRYPTO_CRYPTD compile option is set
  - A crypto daemon which converts an arbitrary synchronous crypto algorithm into an asynchronous algorithm that runs in a kthread

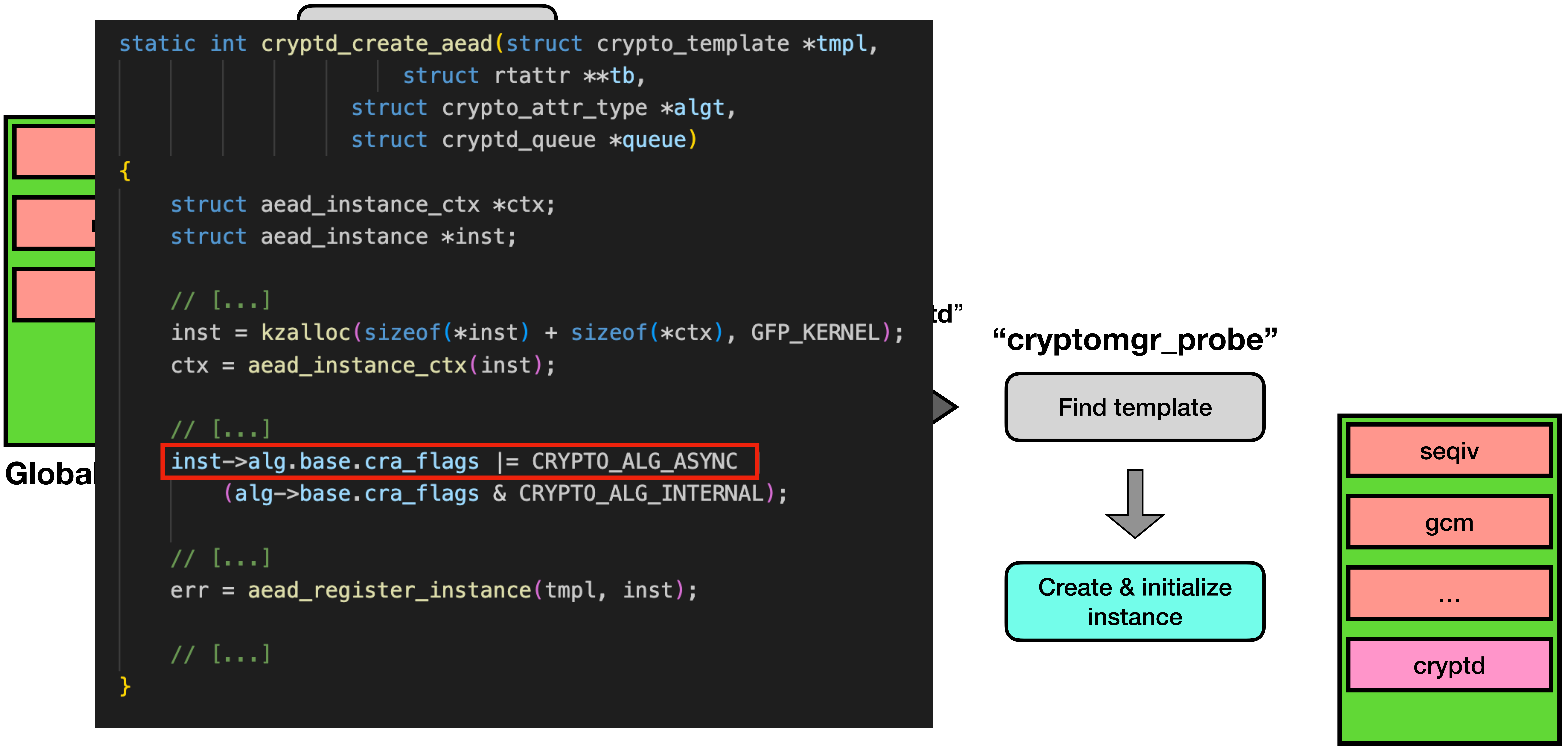
Overview

- **Cryptd**
  - Enabled when the CONFIG_CRYPTO_CRYPTD compile option is set
  - A crypto daemon which converts an arbitrary synchronous crypto algorithm into an asynchronous algorithm that runs in a kthread
  - Used as a **template**

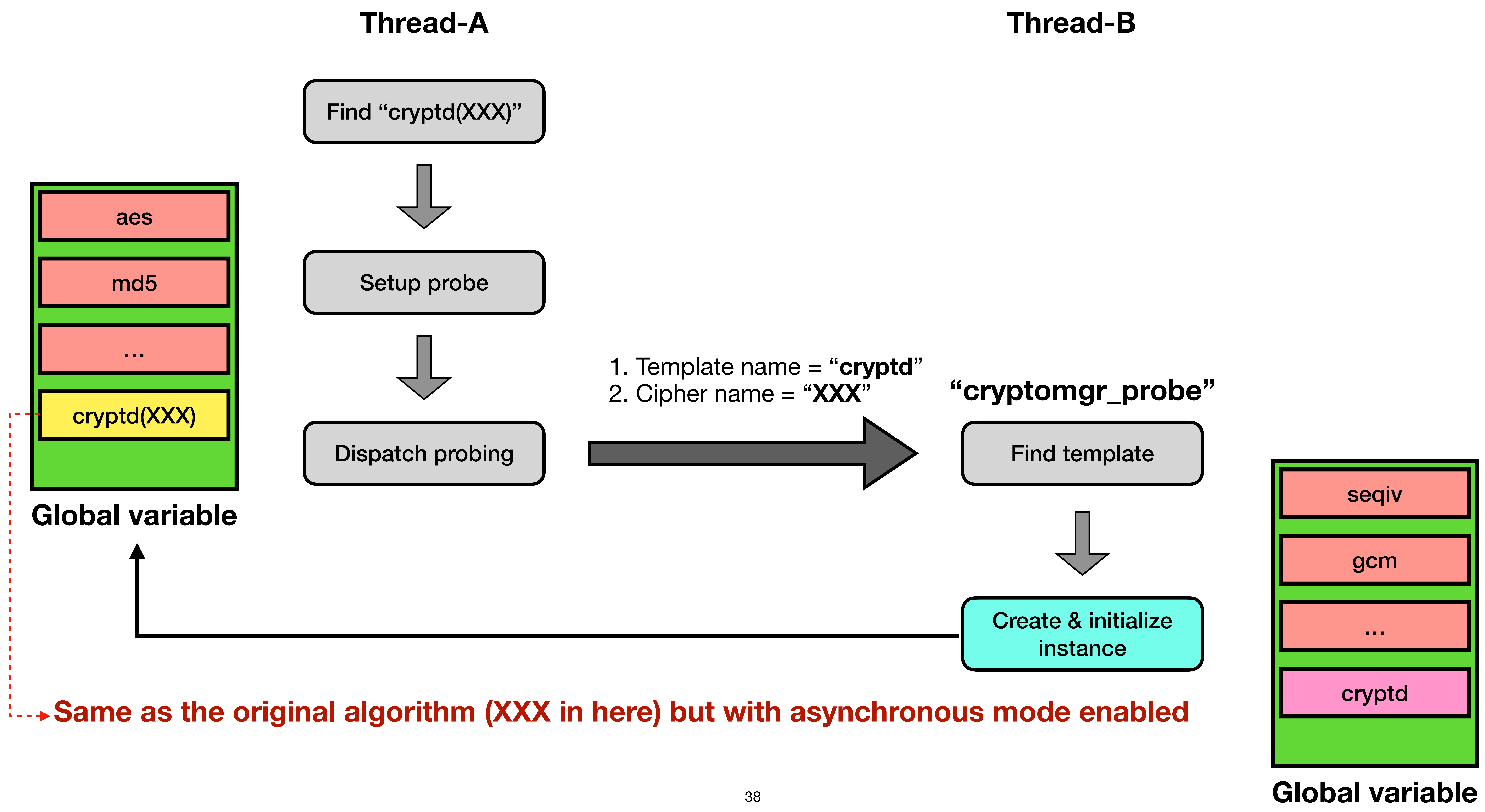

Thread-A
Thread-B
aes
md5
...
Global variable
Find “cryptd(XXX)”
Setup probe
Dispatch probing
1. Template name = “**cryptd**”
2. Cipher name = “**XXX**”
“cryptomgr_probe”
Find template
Create & initialize instance
seqiv
gcm
...
cryptd
Global variable

Thread-A

Thread-B

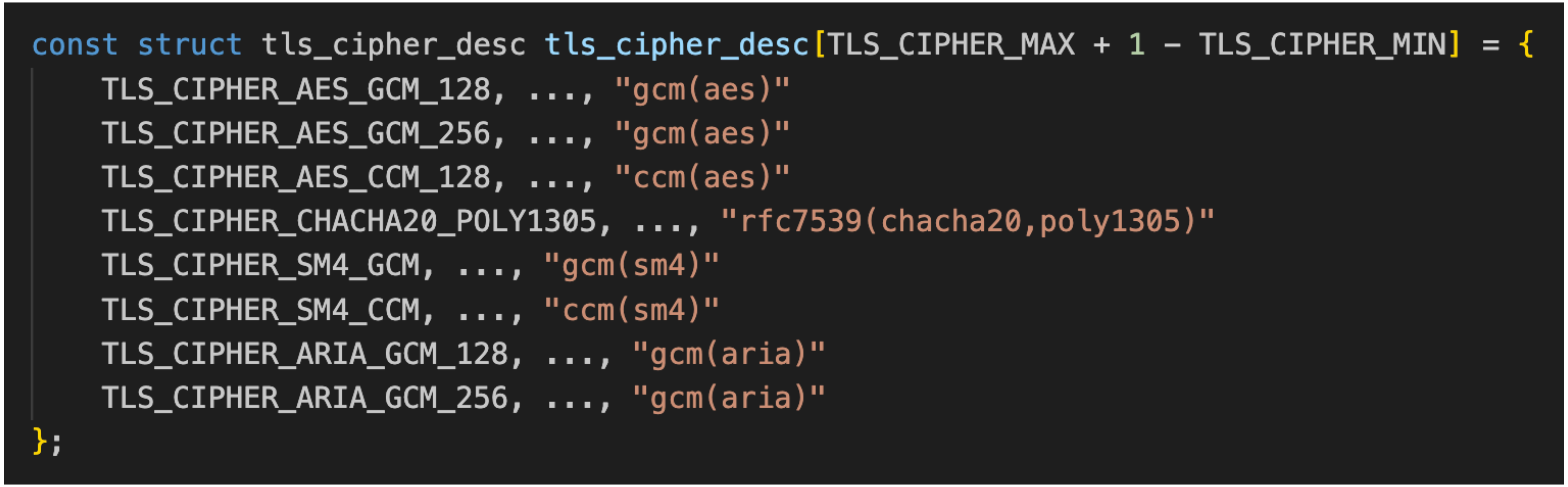


```
static int cryptd_create_aead(struct crypto_template *tmpl,
                              struct rtattr **tb,
                              struct crypto_attr_type *algt,
                              struct cryptd_queue *queue)
{
    struct aead_instance_ctx *ctx;
    struct aead_instance *inst;

    // [...]
    inst = kzalloc(sizeof(*inst) + sizeof(*ctx), GFP_KERNEL);
    ctx = aead_instance_ctx(inst);

    // [...]
    inst->alg.base.cra_flags |= CRYPTO_ALG_ASYNC
        (alg->base.cra_flags & CRYPTO_ALG_INTERNAL);

    // [...]
    err = aead_register_instance(tmpl, inst);

    // [...]
}
```


Thread-A
Thread-B
Find “cryptd(XXX)”
Setup probe
Dispatch probing
1. Template name = “cryptd”
2. Cipher name = “XXX”
“cryptomgr_probe”
Find template
Create & initialize instance
aes
md5
...
cryptd(XXX)
Global variable
seqiv
gcm
...
cryptd
Global variable
Same as the original algorithm (XXX in here) but with asynchronous mode enabled

Overview

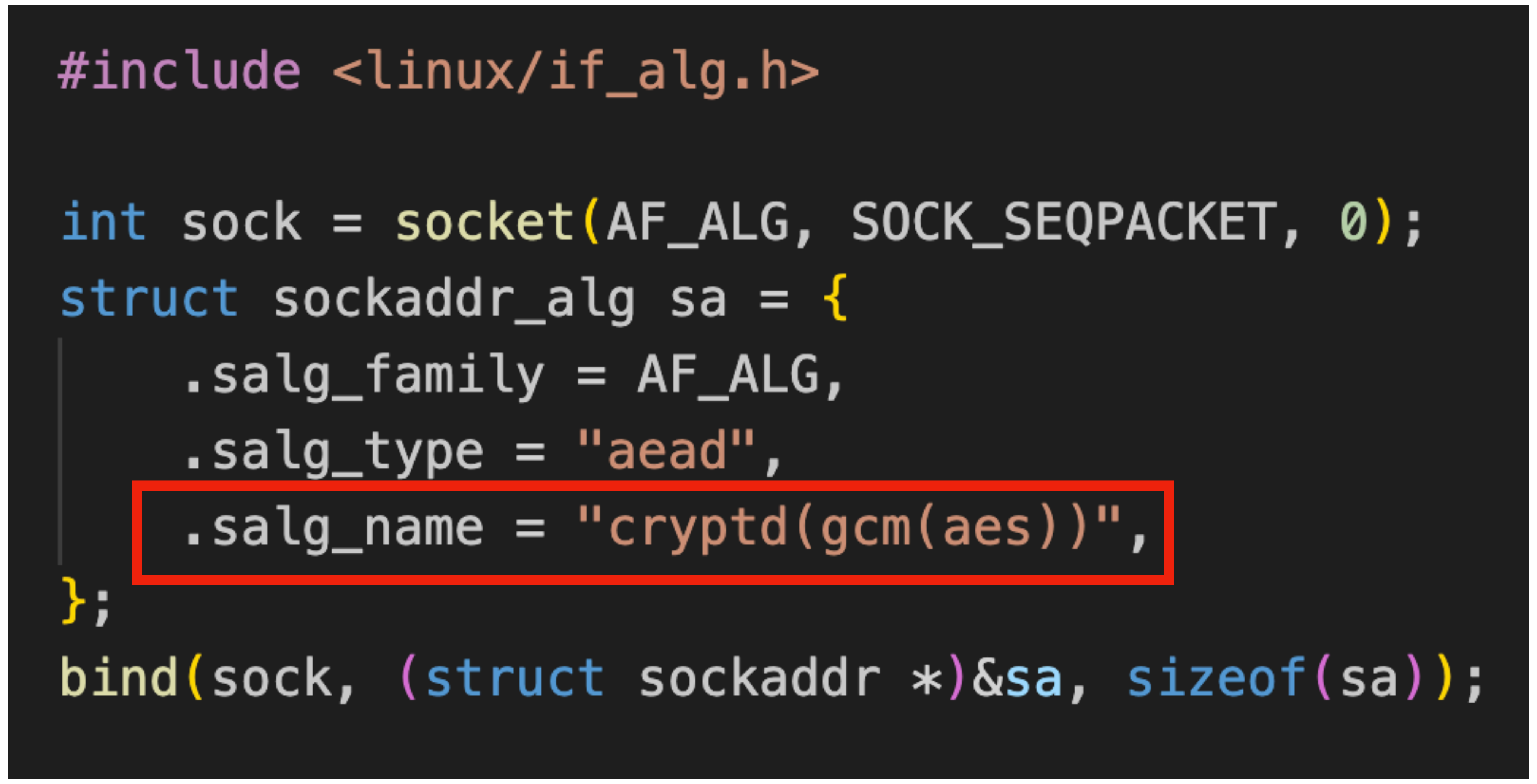
- But how?



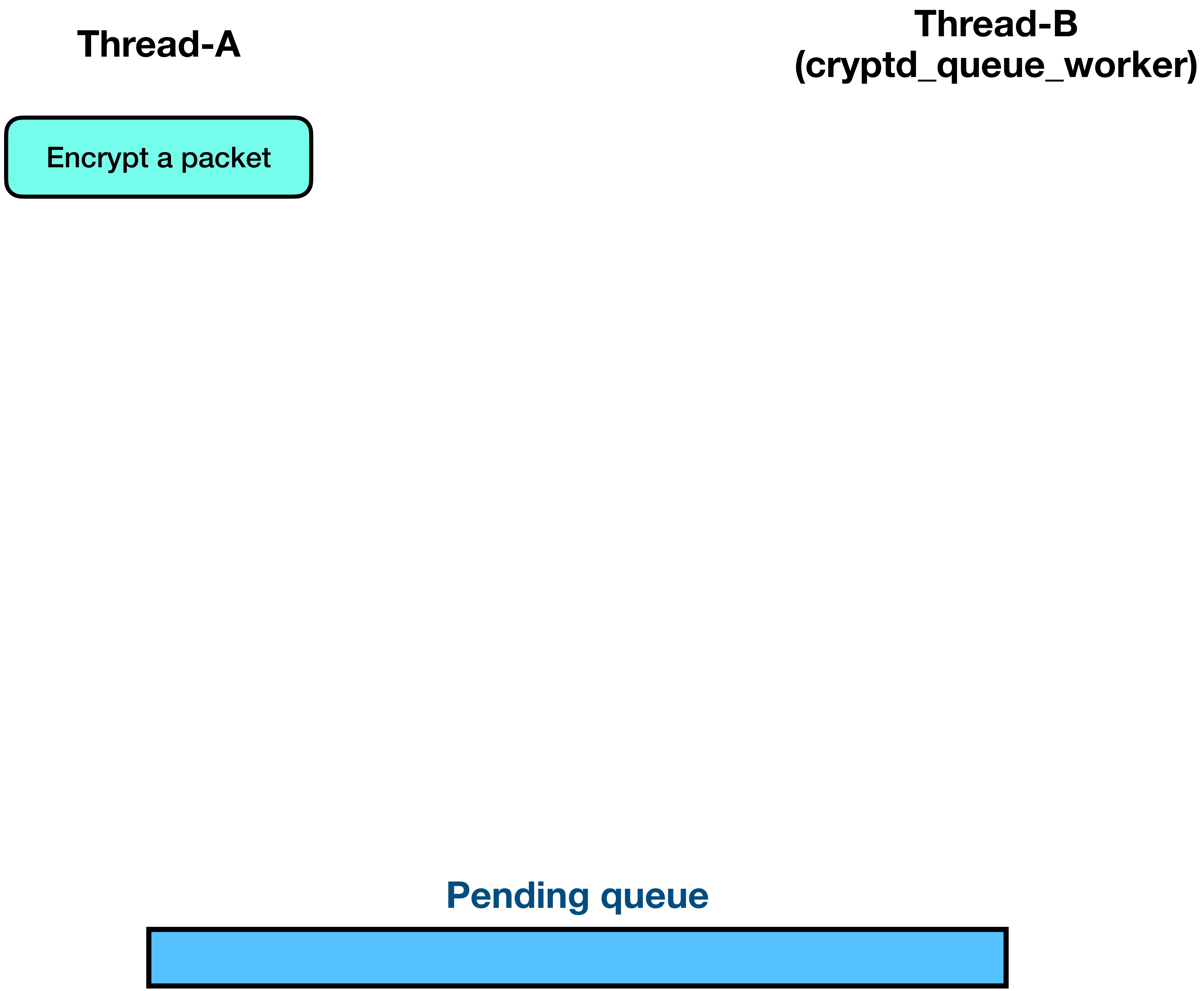
```
const struct tls_cipher_desc tls_cipher_desc[TLS_CIPHER_MAX + 1 - TLS_CIPHER_MIN] = {
    TLS_CIPHER_AES_GCM_128, ..., "gcm(aes)"
    TLS_CIPHER_AES_GCM_256, ..., "gcm(aes)"
    TLS_CIPHER_AES_CCM_128, ..., "ccm(aes)"
    TLS_CIPHER_CHACHA20_POLY1305, ..., "rfc7539(chacha20,poly1305)"
    TLS_CIPHER_SM4_GCM, ..., "gcm(sm4)"
    TLS_CIPHER_SM4_CCM, ..., "ccm(sm4)"
    TLS_CIPHER_ARIA_GCM_128, ..., "gcm(aria)"
    TLS_CIPHER_ARIA_GCM_256, ..., "gcm(aria)"
};
```

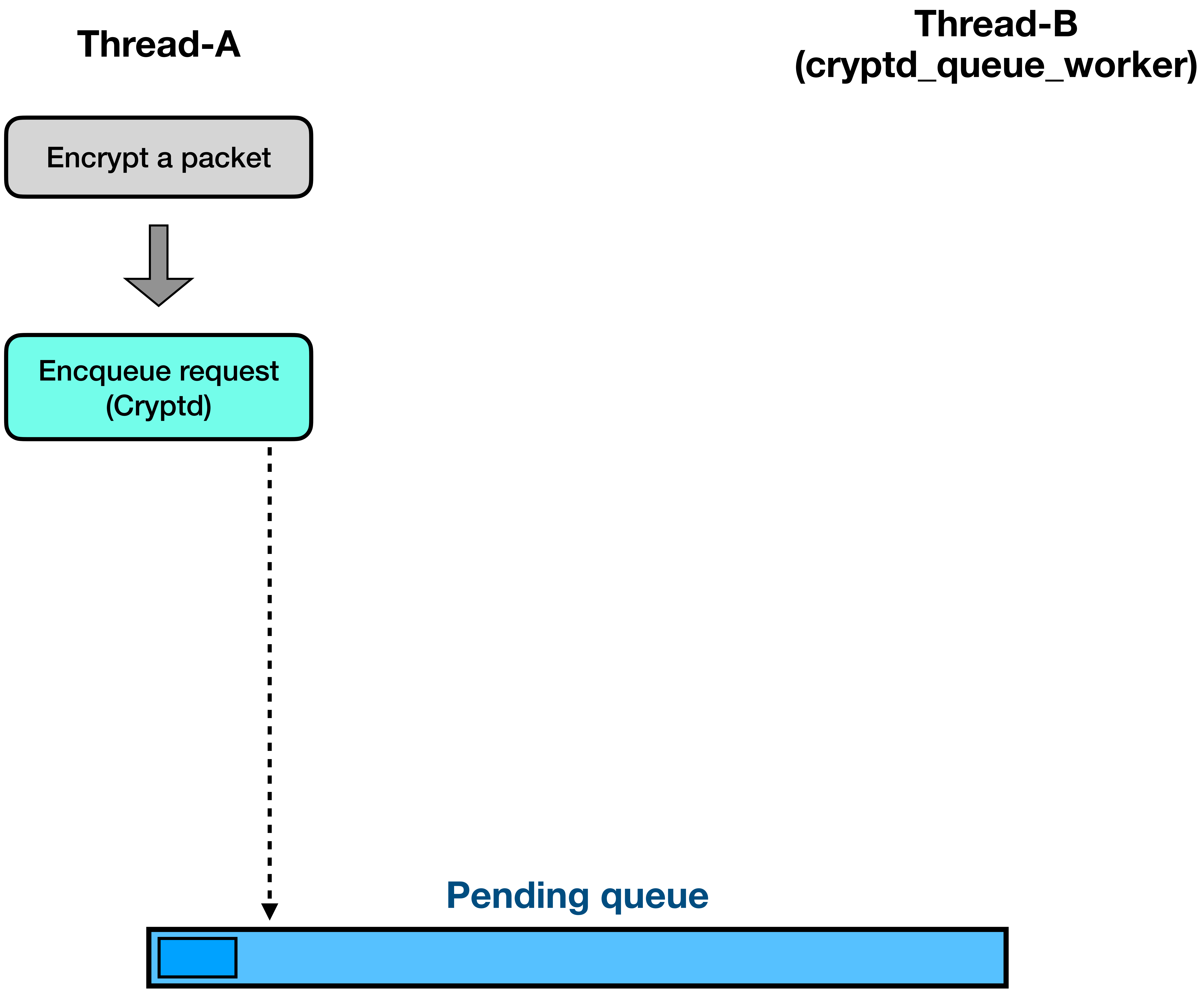
Overview

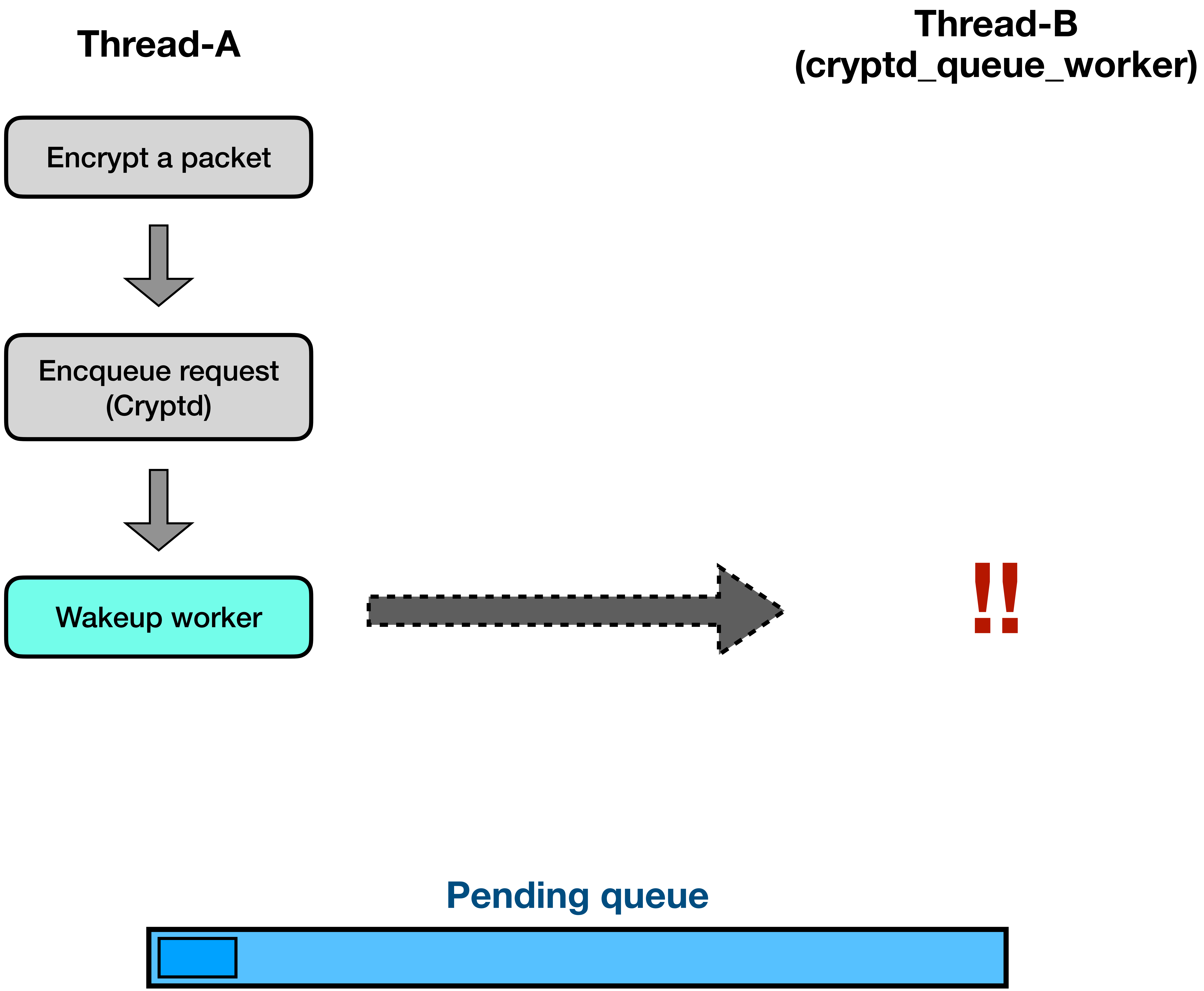
- **AF_ALG**
  - Interface to kernel crypto API
  - Algorithm probing with user-provided algorithm name

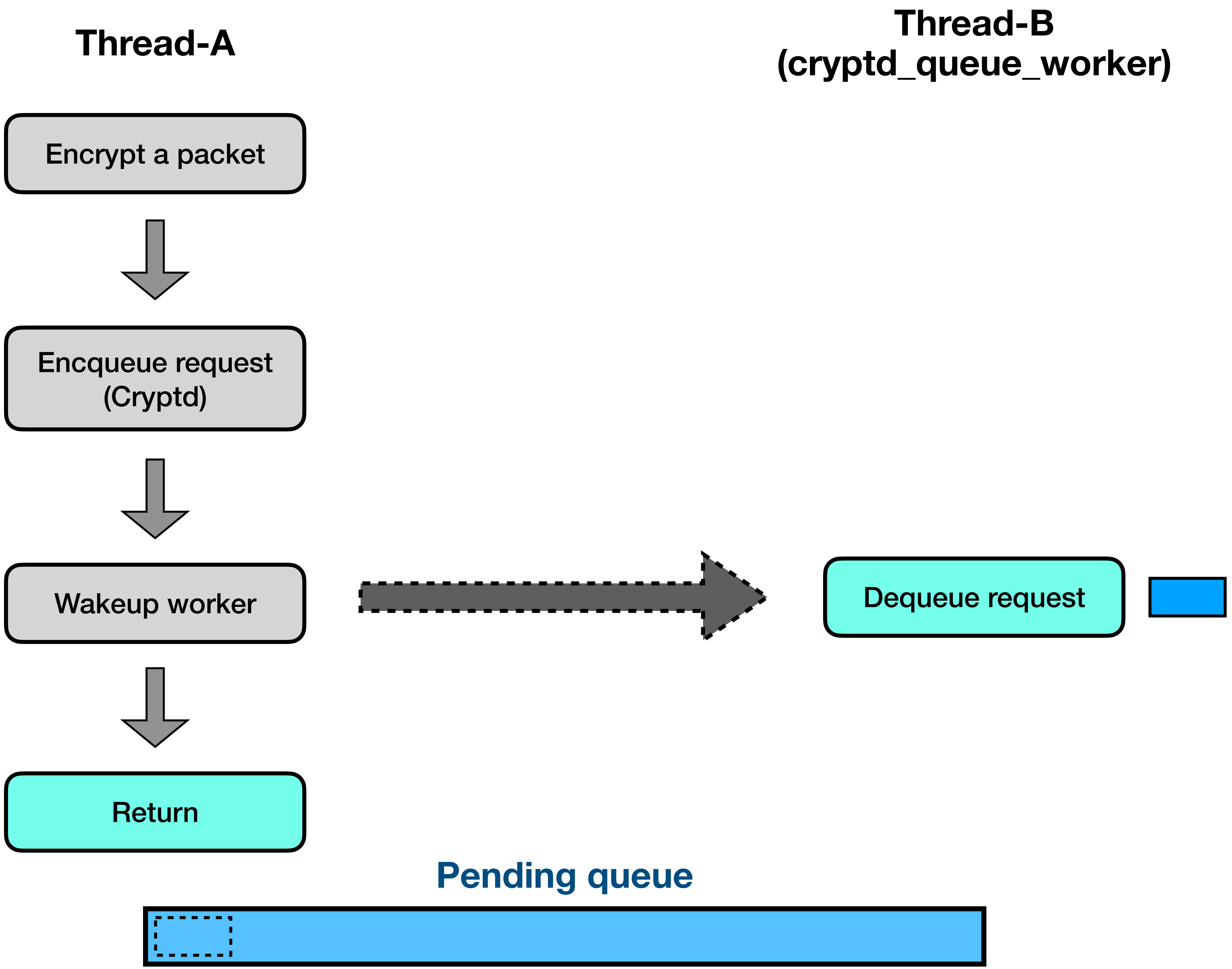


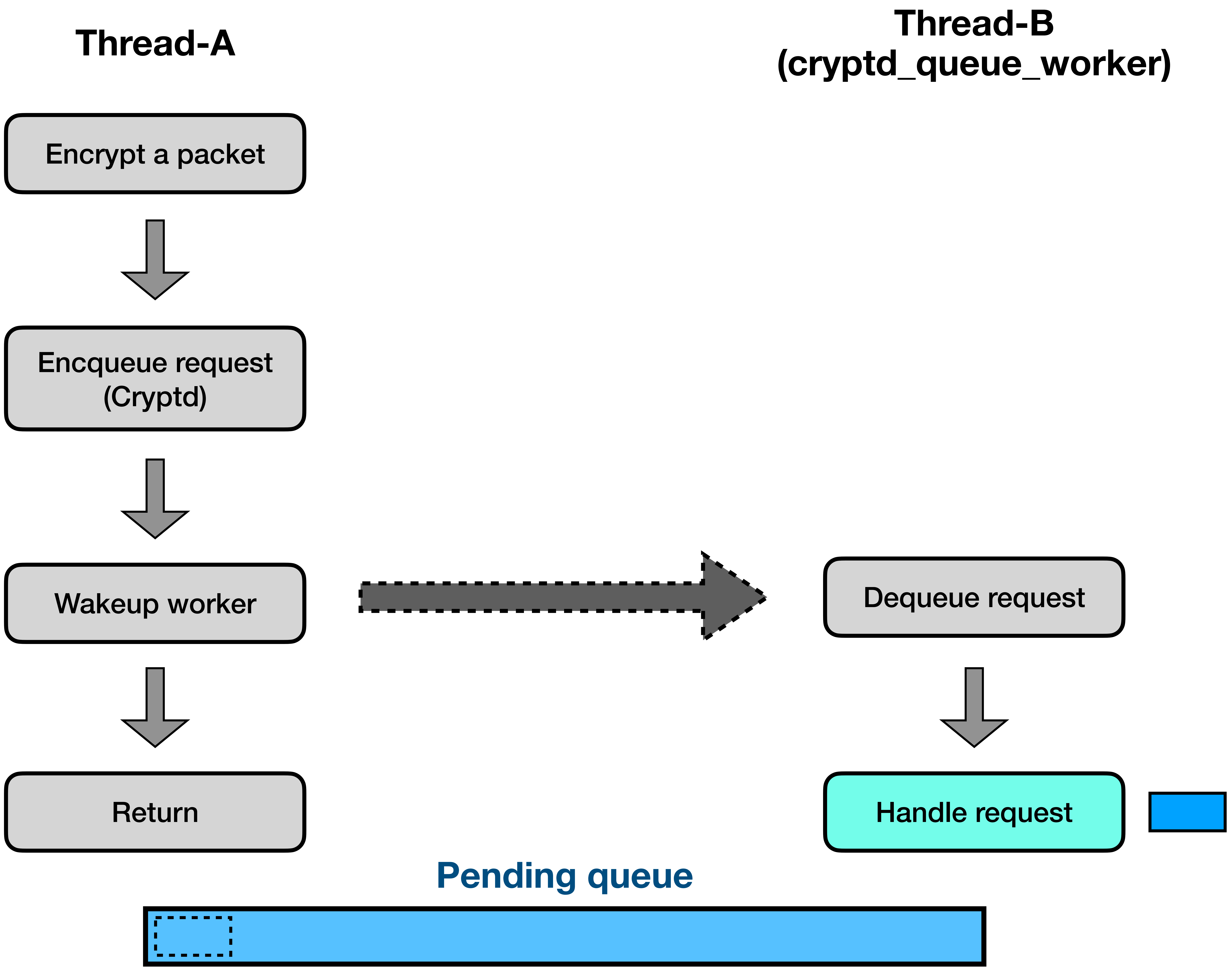
```c
#include <linux/if_alg.h>

int sock = socket(AF_ALG, SOCK_SEQPACKET, 0);
struct sockaddr_alg sa = {
    .salg_family = AF_ALG,
    .salg_type = "aead",
    .salg_name = "cryptd(gcm(aes))",
};
bind(sock, (struct sockaddr *)&sa, sizeof(sa));
```


Thread-A
Thread-B
(cryptd_queue_worker)
Encrypt a packet
Pending queue


Thread-A
Thread-B
(cryptd_queue_worker)
Encrypt a packet
Encqueue request
(Cryptd)
Pending queue

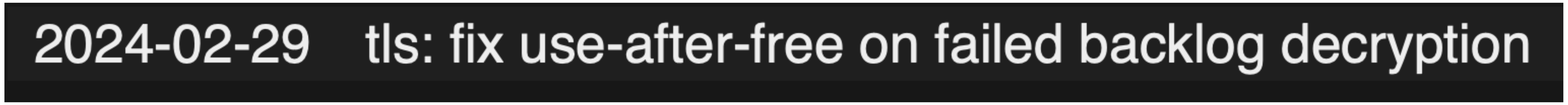

Thread-A
Thread-B
(cryptd_queue_worker)
Encrypt a packet
Encqueue request
(Cryptd)
Wakeup worker
!!
Pending queue


Thread-A
Thread-B
(cryptd_queue_worker)
Encrypt a packet
Encqueue request
(Cryptd)
Wakeup worker
Return
Dequeue request
Pending queue


Thread-A
Thread-B
(cryptd_queue_worker)
Encrypt a packet
Encqueue request (Cryptd)
Wakeup worker
Return
Dequeue request
Handle request
Pending queue

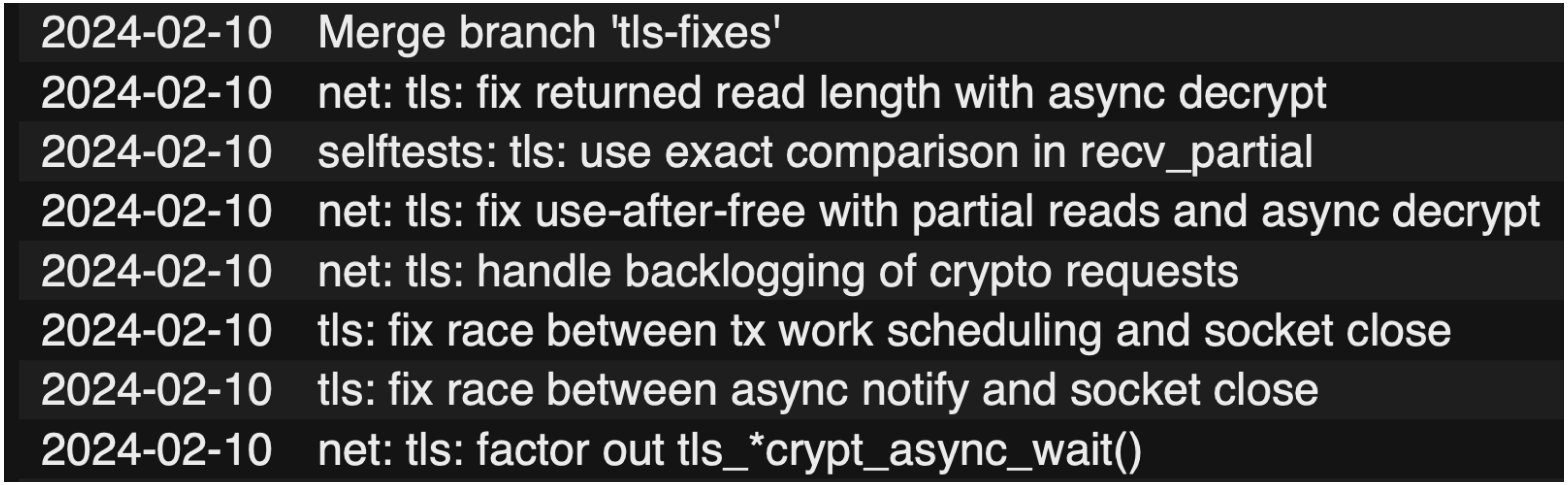
Vulnerability

Vulnerability



CVE-2024-26800

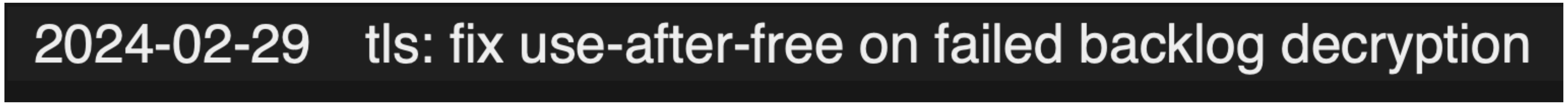
```
2024-02-29   tls: fix use-after-free on failed backlog decryption
```



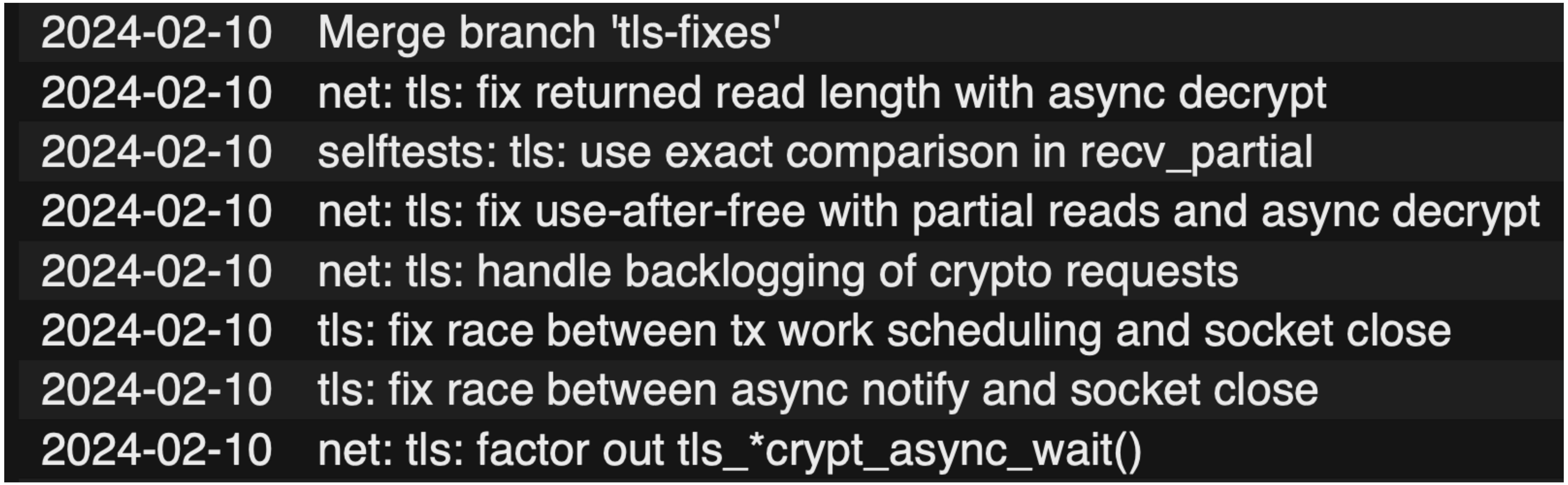
```
2024-02-10   Merge branch 'tls-fixes'
2024-02-10   net: tls: fix returned read length with async decrypt
2024-02-10   selftests: tls: use exact comparison in recv_partial
2024-02-10   net: tls: fix use-after-free with partial reads and async decrypt
2024-02-10   net: tls: handle backlogging of crypto requests
2024-02-10   tls: fix race between tx work scheduling and socket close
2024-02-10   tls: fix race between async notify and socket close
2024-02-10   net: tls: factor out tls_*crypt_async_wait()
```

CVE-2024-26582
CVE-2024-26584
CVE-2024-26583
CVE-2024-26585

Vulnerability



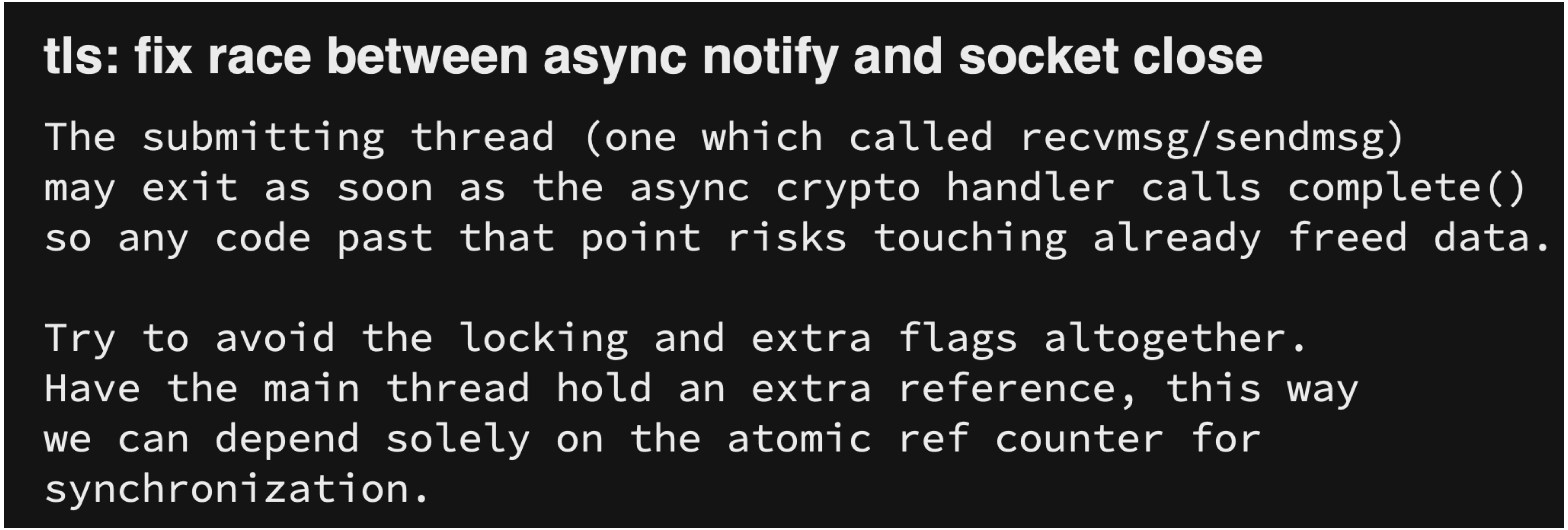
```
2024-02-29   tls: fix use-after-free on failed backlog decryption
```



```
2024-02-10   Merge branch 'tls-fixes'
2024-02-10   net: tls: fix returned read length with async decrypt
2024-02-10   selftests: tls: use exact comparison in recv_partial
2024-02-10   net: tls: fix use-after-free with partial reads and async decrypt
2024-02-10   net: tls: handle backlogging of crypto requests
2024-02-10   tls: fix race between tx work scheduling and socket close
2024-02-10   tls: fix race between async notify and socket close
2024-02-10   net: tls: factor out tls_*crypt_async_wait()
```

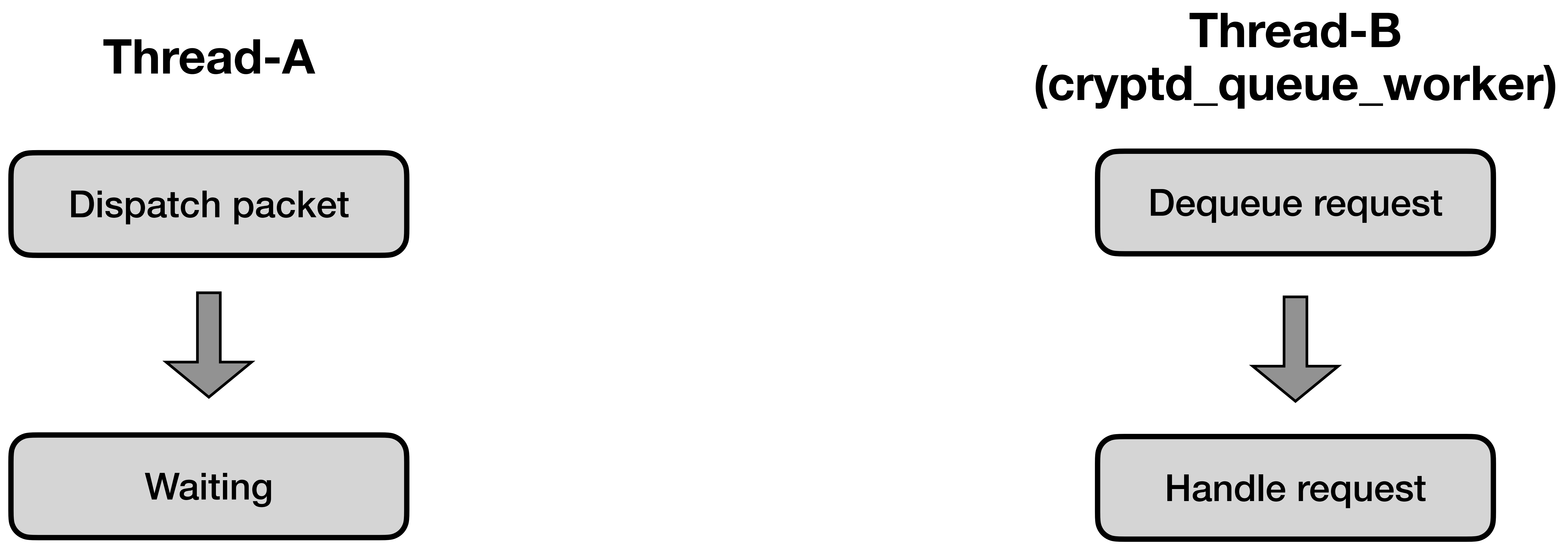
CVE-2024-26585

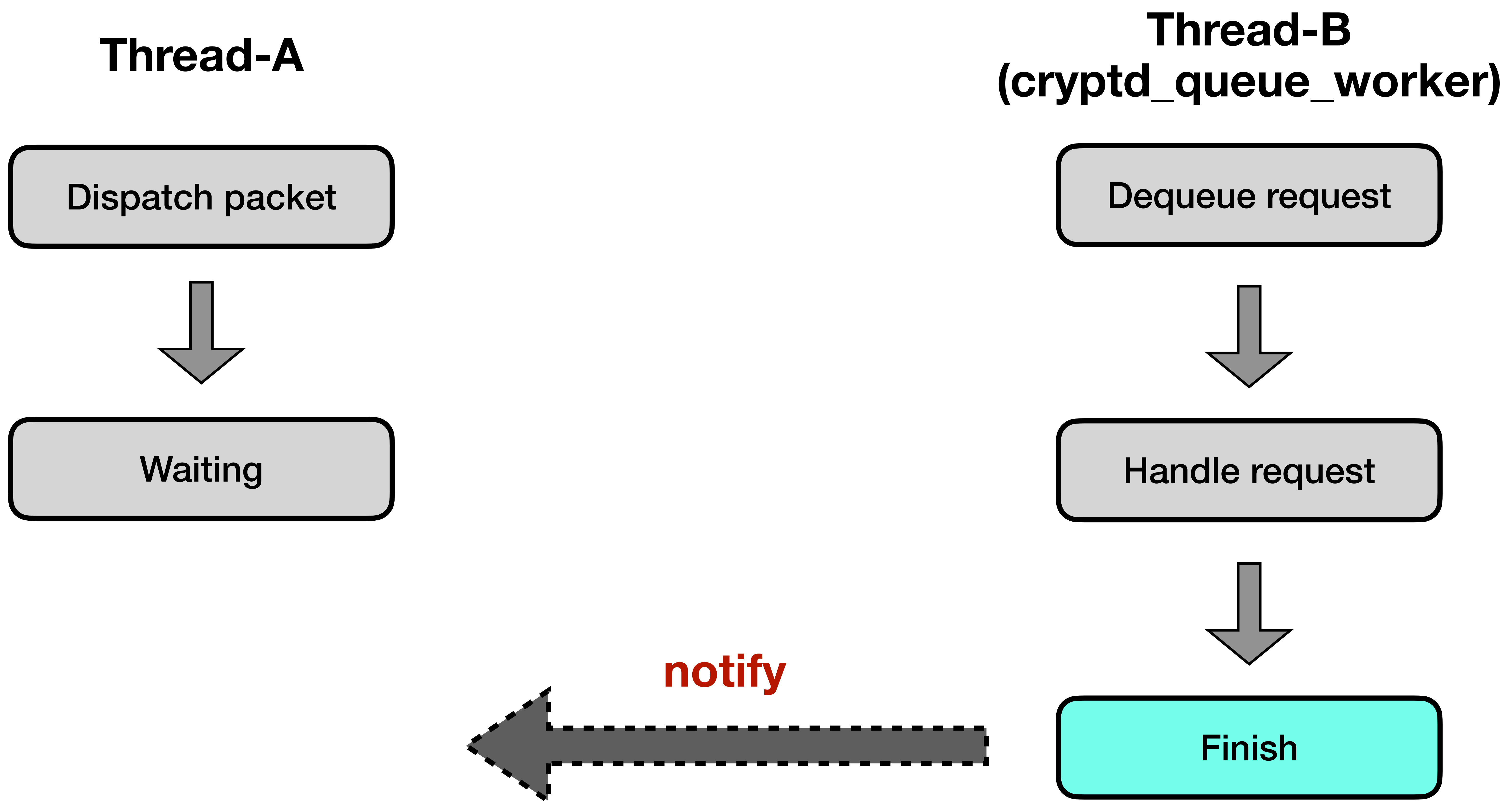
Vulnerability
CVE-2024-26585

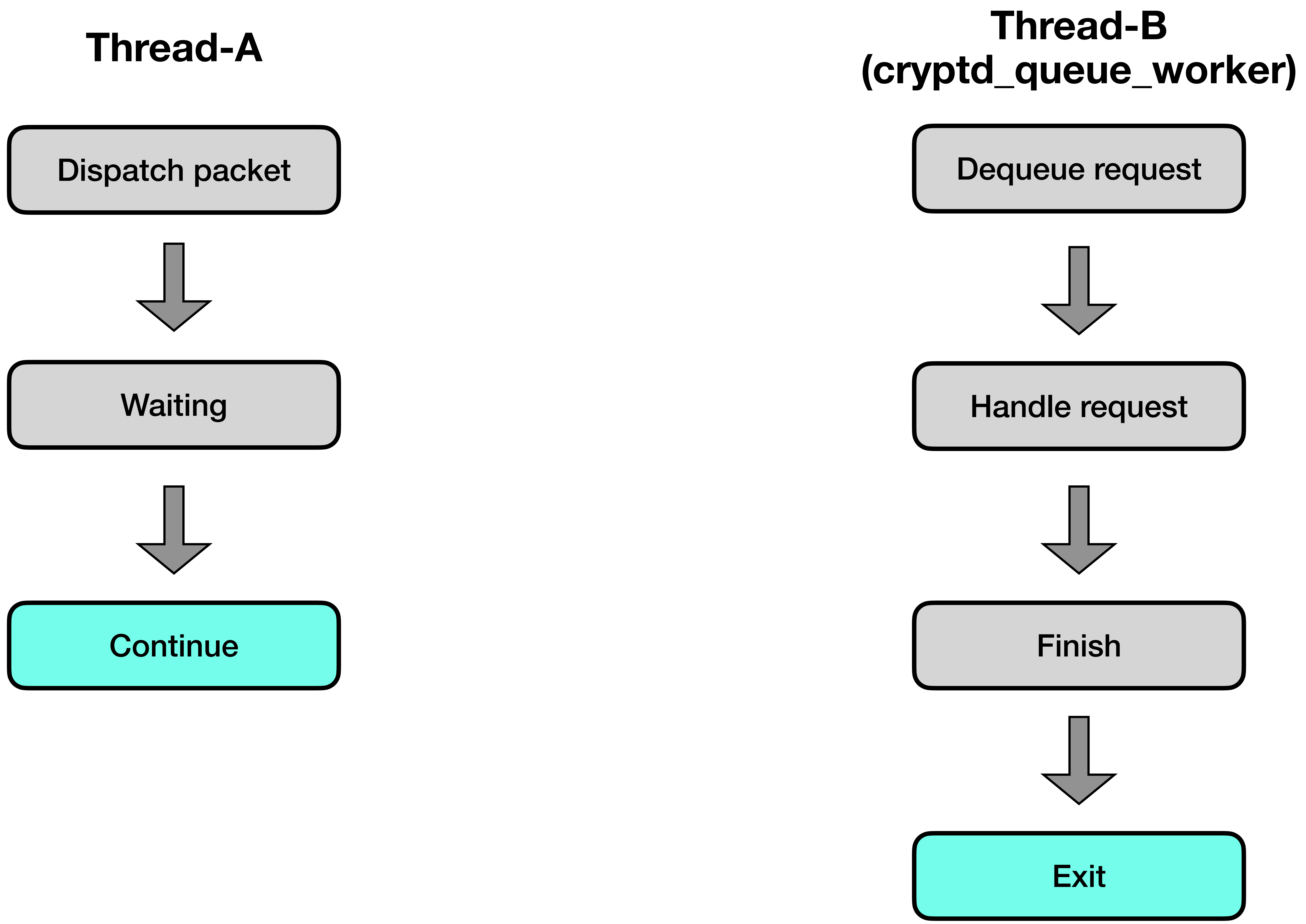


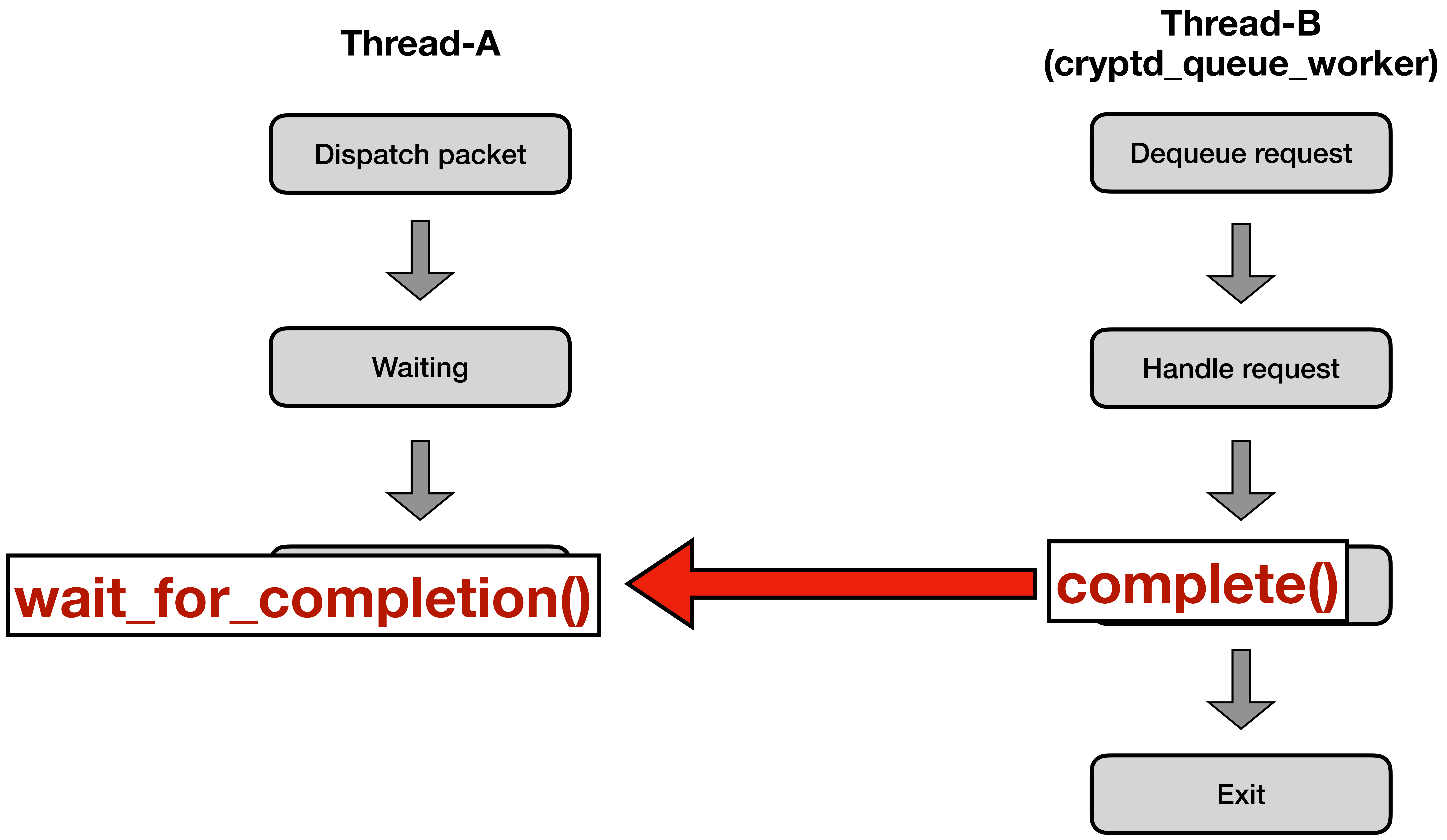
tls: fix race between async notify and socket close

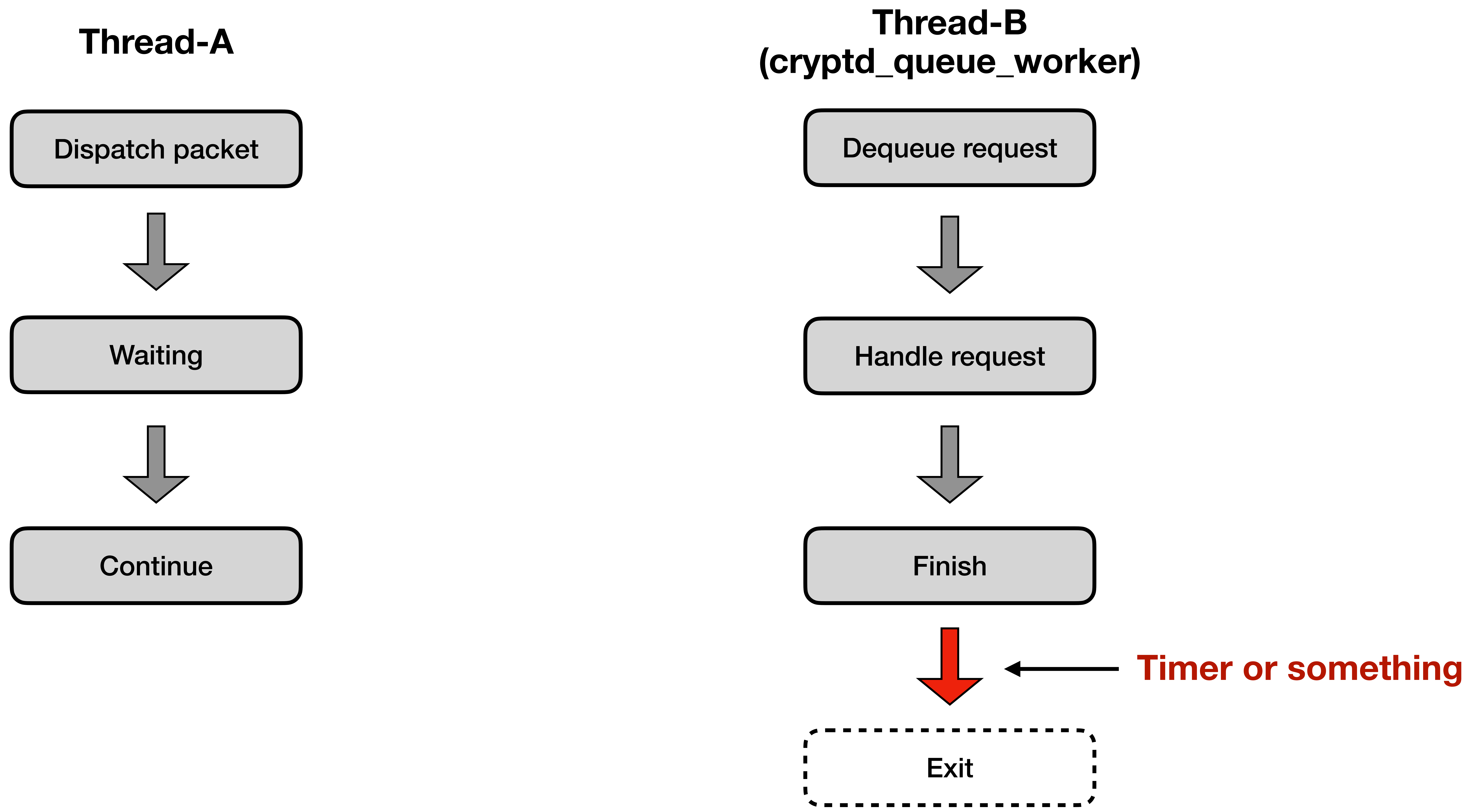
```
The submitting thread (one which called recvmsg/sendmsg)
may exit as soon as the async crypto handler calls complete()
so any code past that point risks touching already freed data.

Try to avoid the locking and extra flags altogether.
Have the main thread hold an extra reference, this way
we can depend solely on the atomic ref counter for
synchronization.
```

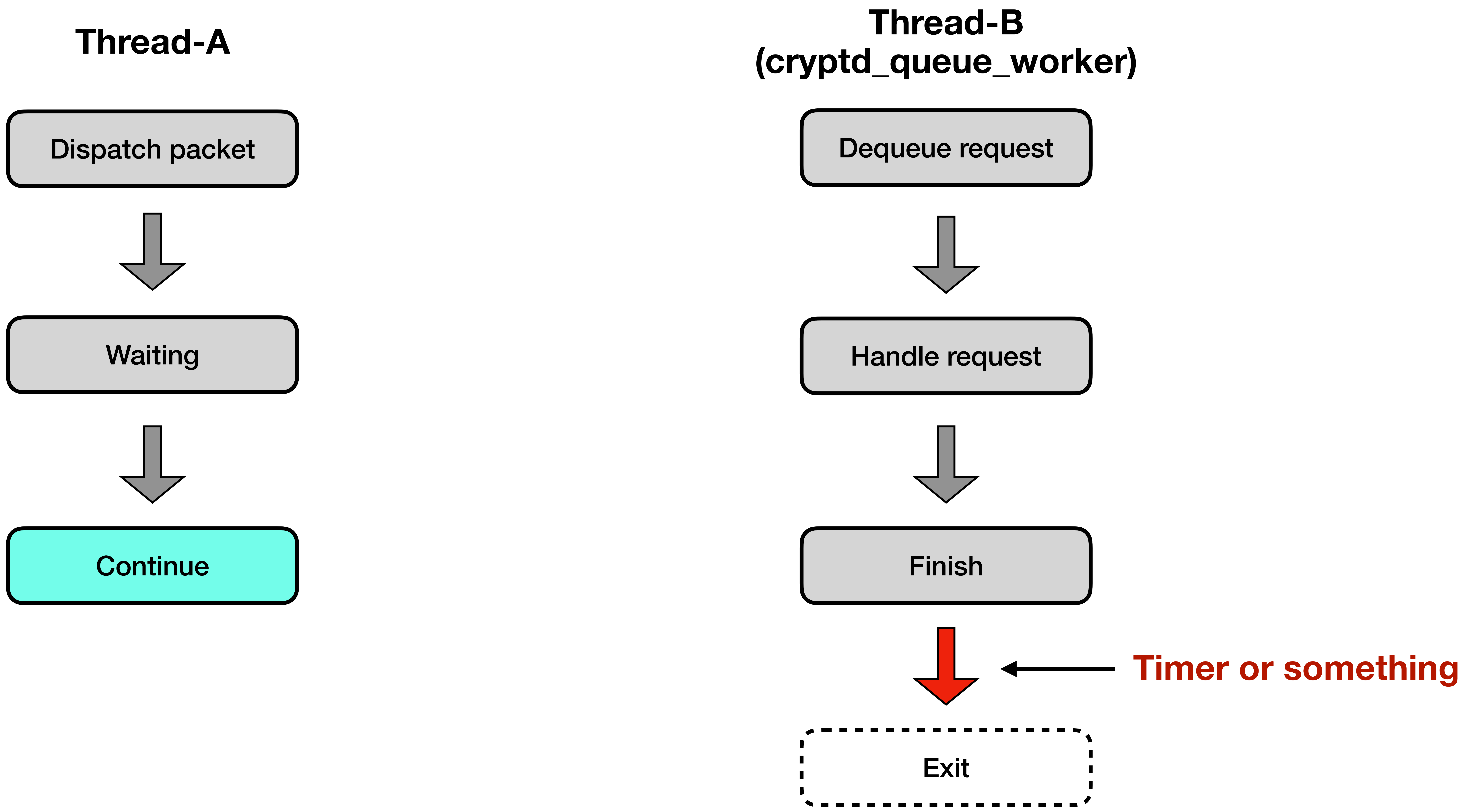

Thread-A
Dispatch packet
Waiting
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request

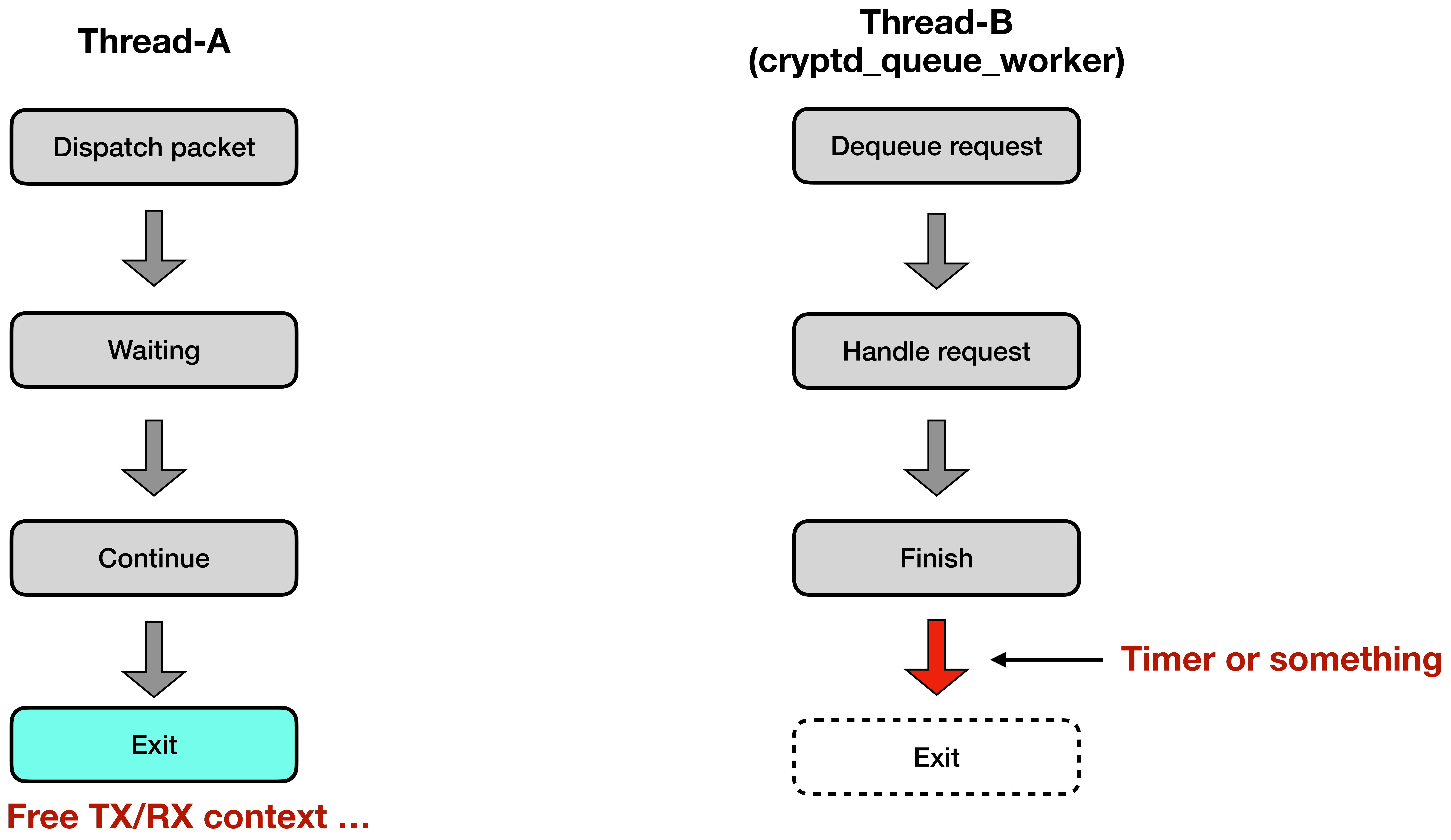

Thread-A
Thread-B
(cryptd_queue_worker)
Dispatch packet
Waiting
Dequeue request
Handle request
Finish
notify

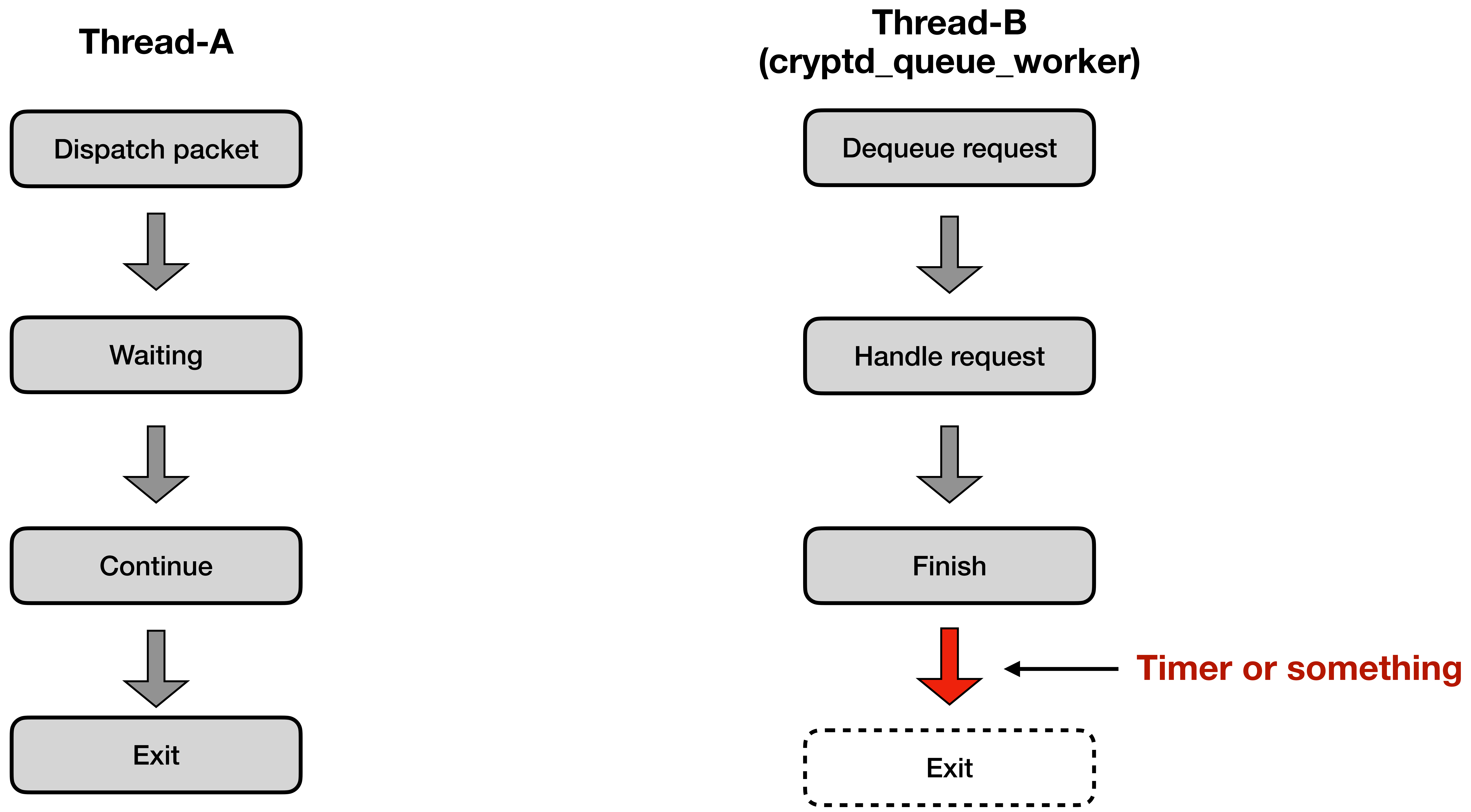

Thread-A
Dispatch packet
Waiting
Continue
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request
Finish
Exit

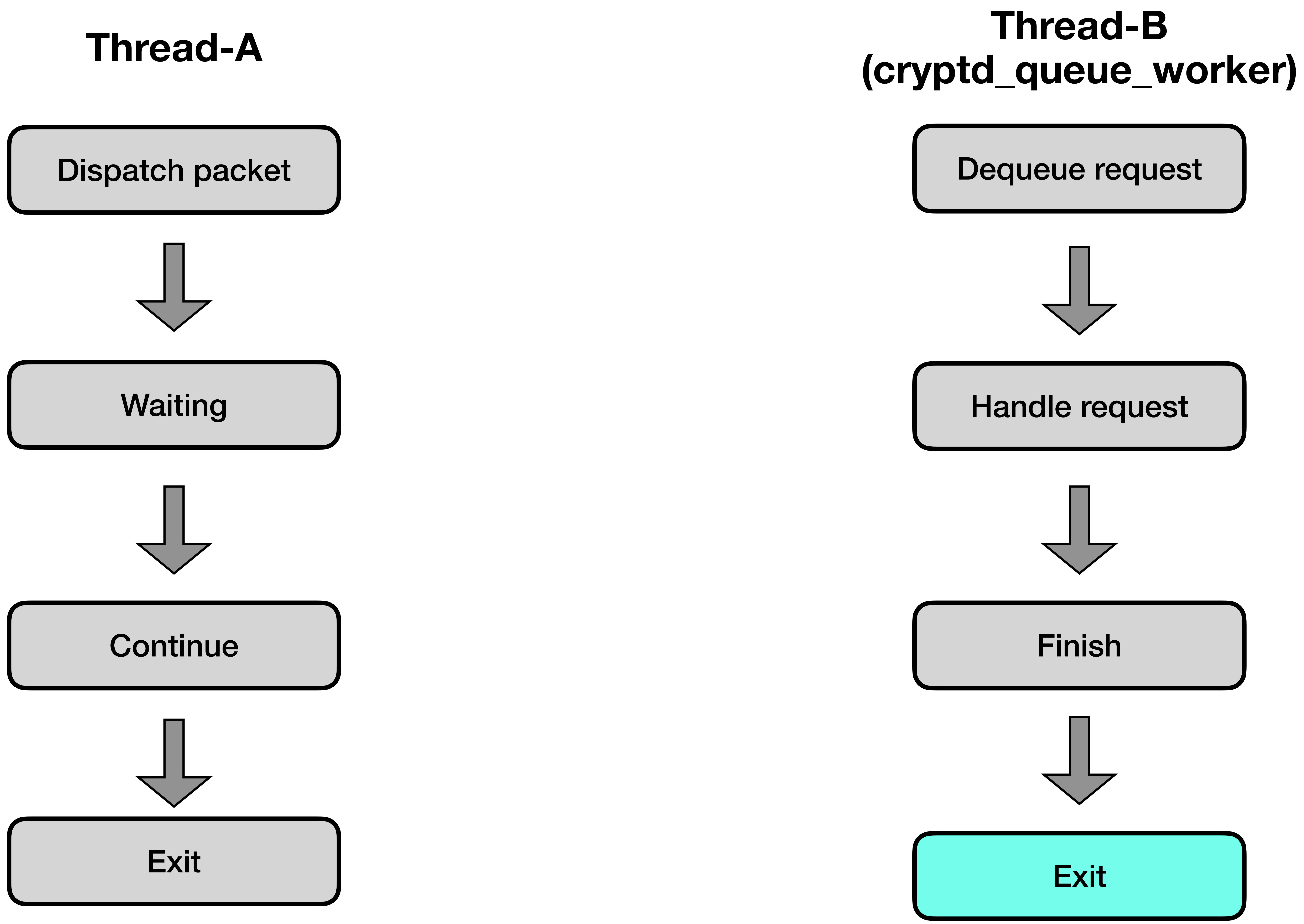

Thread-A
Thread-B
(cryptd_queue_worker)
Dispatch packet
Waiting
wait_for_completion()
Dequeue request
Handle request
complete()
Exit

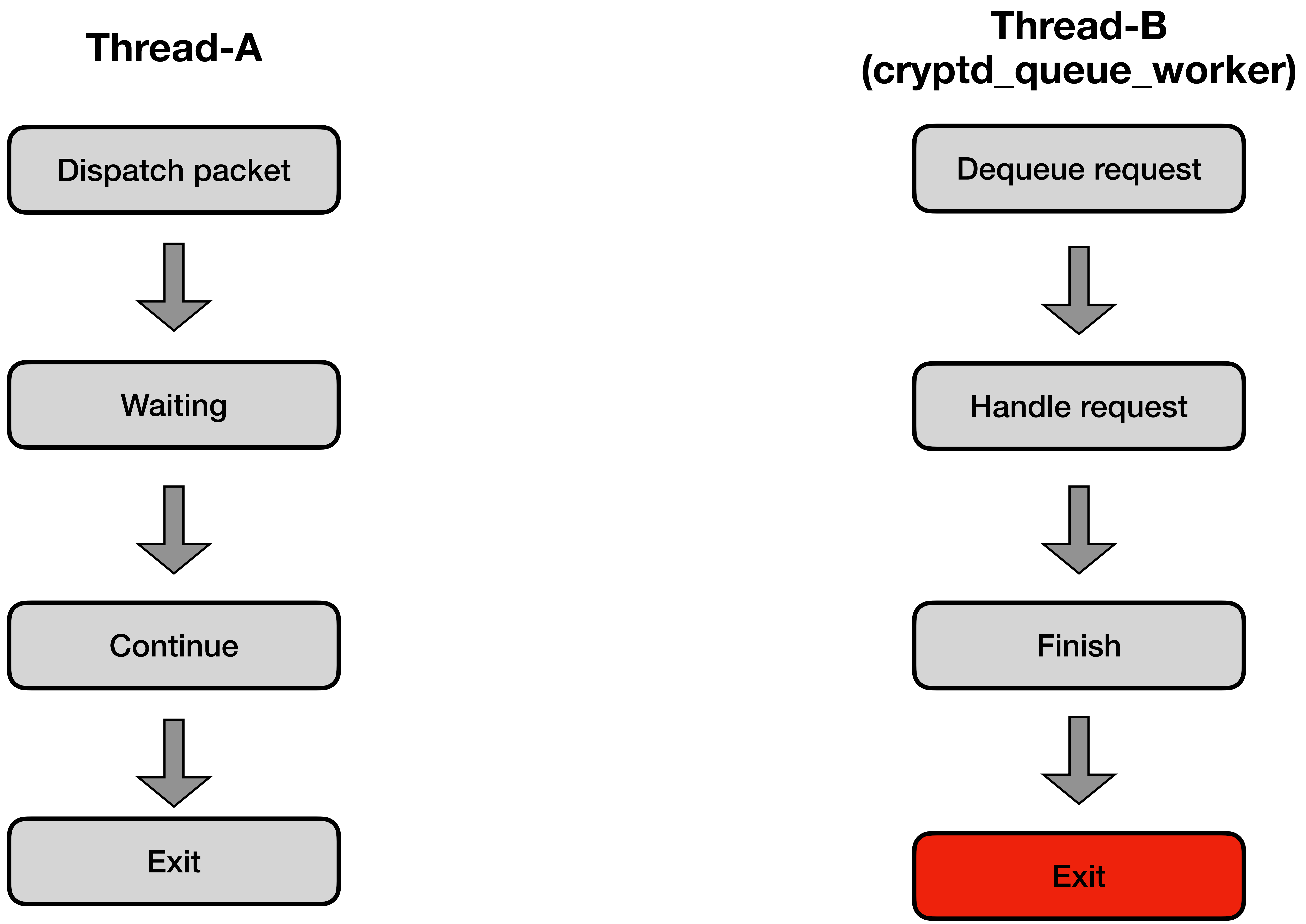

Thread-A
Dispatch packet
Waiting
Continue
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request
Finish
Timer or something
Exit


Thread-A
Dispatch packet
Waiting
Continue
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request
Finish
Timer or something
Exit


Thread-A
Dispatch packet
Waiting
Continue
Exit
Free TX/RX context …
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request
Finish
Timer or something
Exit


Thread-A
Dispatch packet
Waiting
Continue
Exit
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request
Finish
Timer or something
Exit


Thread-A
Dispatch packet
Waiting
Continue
Exit
Thread-B
(cryptd_queue_worker)
Dequeue request
Handle request
Finish
Exit



UAF when accessing TX/RX context object