RAKIS: Secure Fast I/O Primitives Across Trust Boundaries on Intel SGX

Mansour Alharthi, Fan Sang, Dmitrii Kuvaiskii, Mona Vij and Taesoo Kim

SSLab @GeorgiaTech

intel

Georgia Tech

Trusted Execution Environments (TEEs)

- TEEs offers a secure execution environment for applications.
- Intel SGX, introduced in 2015, still sees applications today, particularly in cloud computing.
- Intel SGX offers lightweight TEE with encrypted enclaves.
  - Applications enter enclaves, where execution is secure - even from privileged entities like the OS.
  - Applications exit enclaves and go back to normal execution.

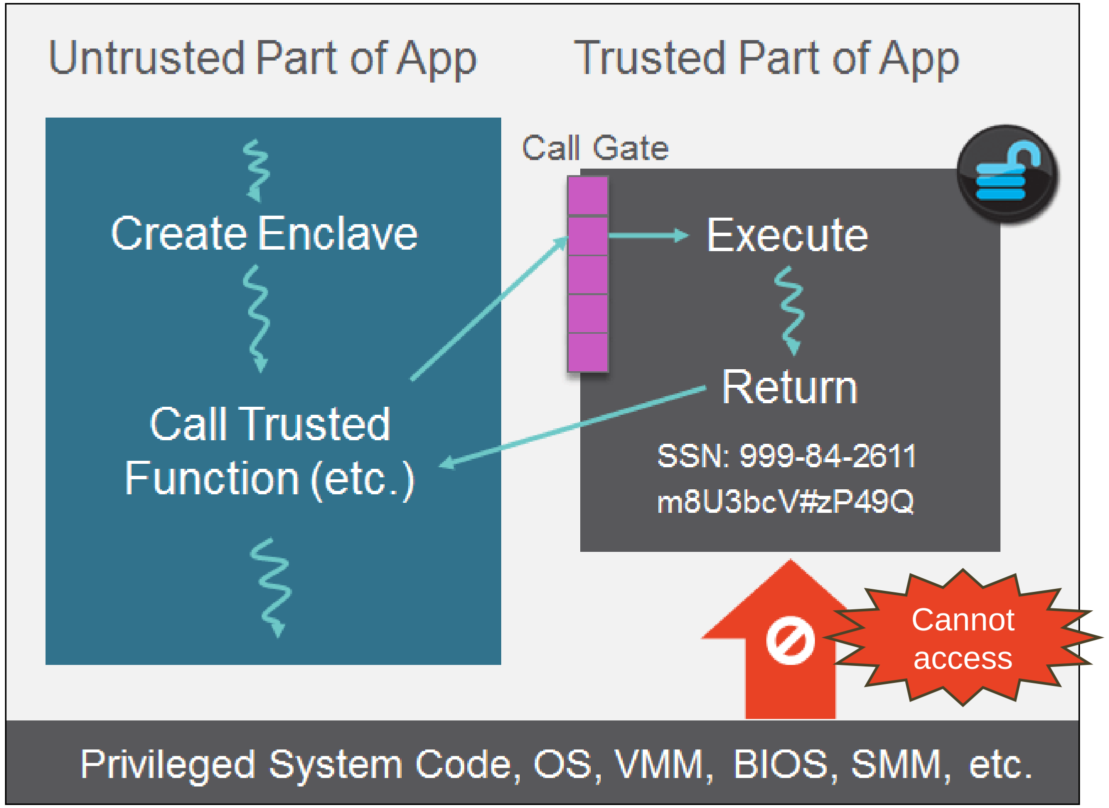


Figure credit: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sgx-web-based-training.html

Intel SGX – Enclave Programs

- Enclave programs have restricted access to OS services.

- To make a syscall, the enclave program must:
  1. Copy syscall data to untrusted memory.
  2. Exit the encalve.
  3. Perform the syscall outside the enclave.
  4. Re-enter the enclave and copy the result inside.

- This means significant cost for IO operations.

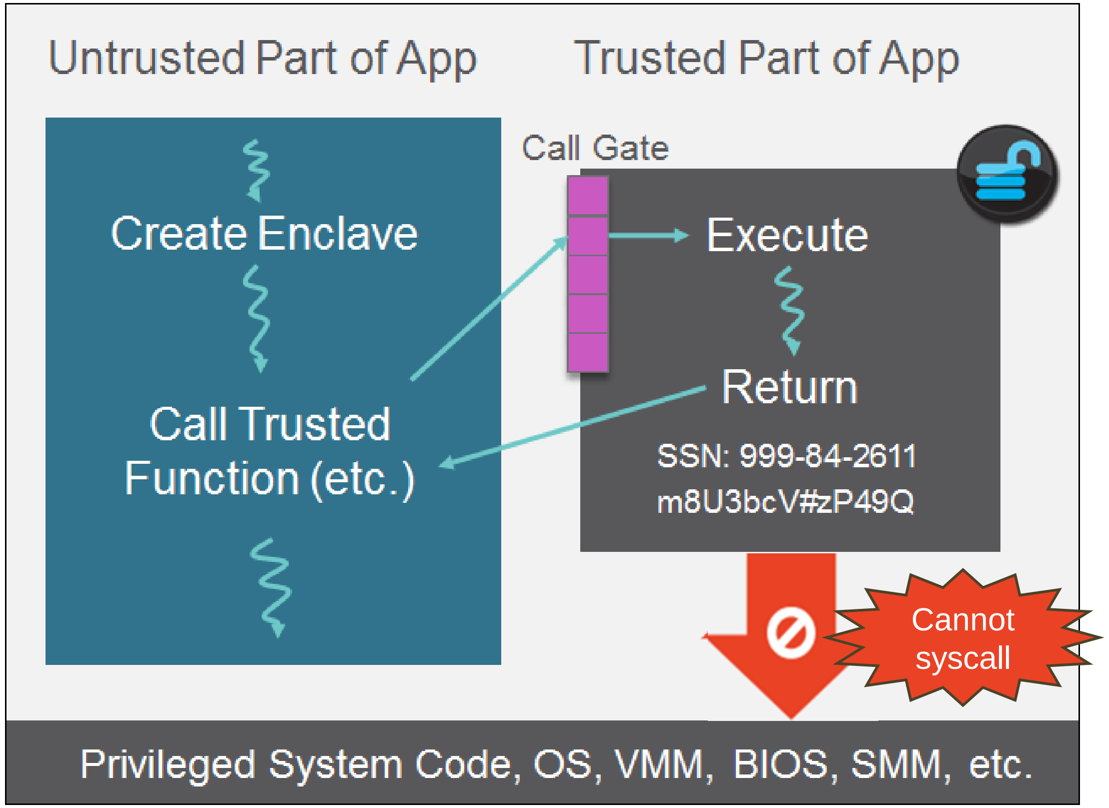


Figure credit: https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sgx-web-based-training.html

Intel SGX - I/O cost

- Enclave entry/exit costs a minimum of 8200 CPU cycles; not including data transfer cost [1].
  - This context switch cost is applied to all IO calls, i.e. per every user `send()`/`recv()` call.

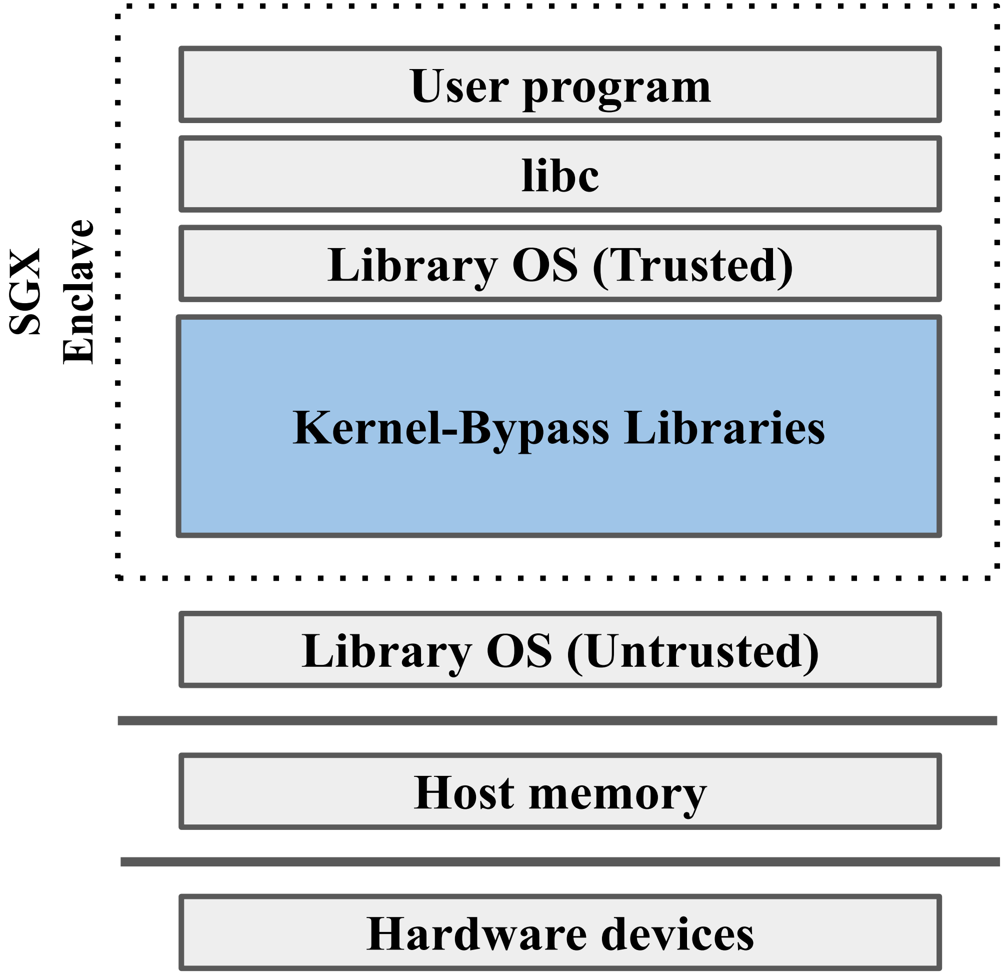
- Our experiments show that I/O-intensive programs can run up to 5x slower inside SGX enclaves.
  - The main cause is the need to exit the enclave and reenter per IO syscall.

[1]: Ofir Weisse, Valeria Bertacco, and Todd Austin. 2017. Regaining Lost Cycles with HotCalls: A Fast Interface for SGX Secure Enclaves.

State-of-the-art: Direct I/O inside SGX

- Utilize kernel-bypass libraries inside SGX enclaves.

- Limitations of this approach:
  1. Significant increase in TCB size.
     - ➢ More attack surface & security risks.
  2. Difficulty in deployment.
     - ➢ Limits adoption and increases compatibility challenges.
  3. Inclusion of unnecessary components.
     - ➢ Requires heavy OS features like thread scheduling, which are unnecessary for lightweight enclave programs.



RAKIS - Goals

- Enable fast I/O primitives inside SGX enclaves that:
    1. Maintains the security guarantees of Intel SGX.
    2. Minimal increase in TCB size.
    3. Run unmodified user programs.

- To achieve its goals, RAKIS leverages two recently introduced Linux kernel I/O primitives:
    1. eXpress Data Path (XDP).
    2. io_uring.

Fast I/O Kernel Primitives (FIOKPs)

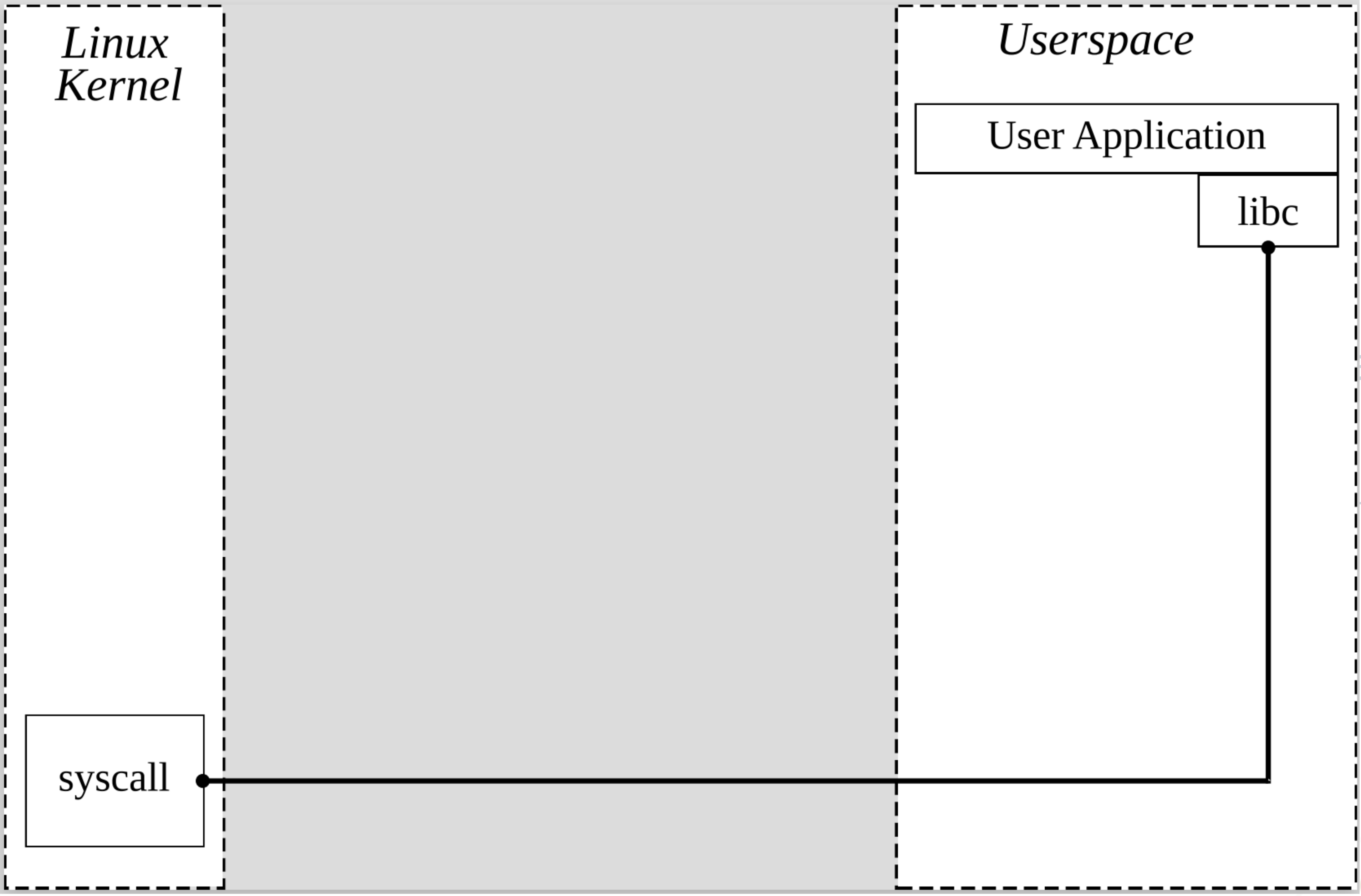
- **eXpress Data Path (XDP):** Enables high-performance packet processing at the earliest point in the Linux kernel.

- **io_uring:** Enables efficient asynchronous I/O operations.

- FIOKPs:
  - Enhance I/O performance by reducing system call overhead.
  - Utilize shared memory and ring buffers for operations.

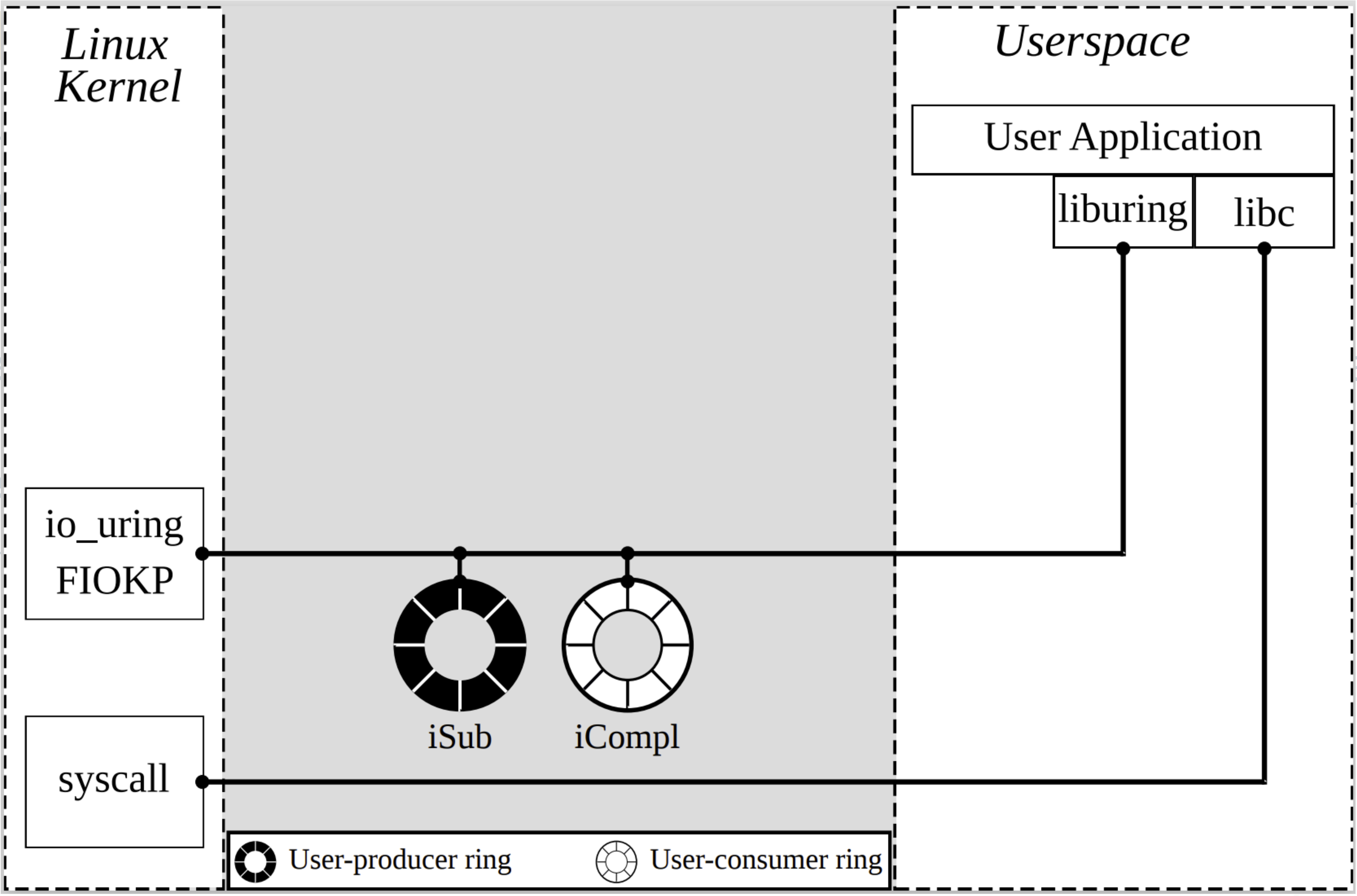
Fast I

- Recen
  - eXp
  - io_u
- These ations.
- They e th syscal



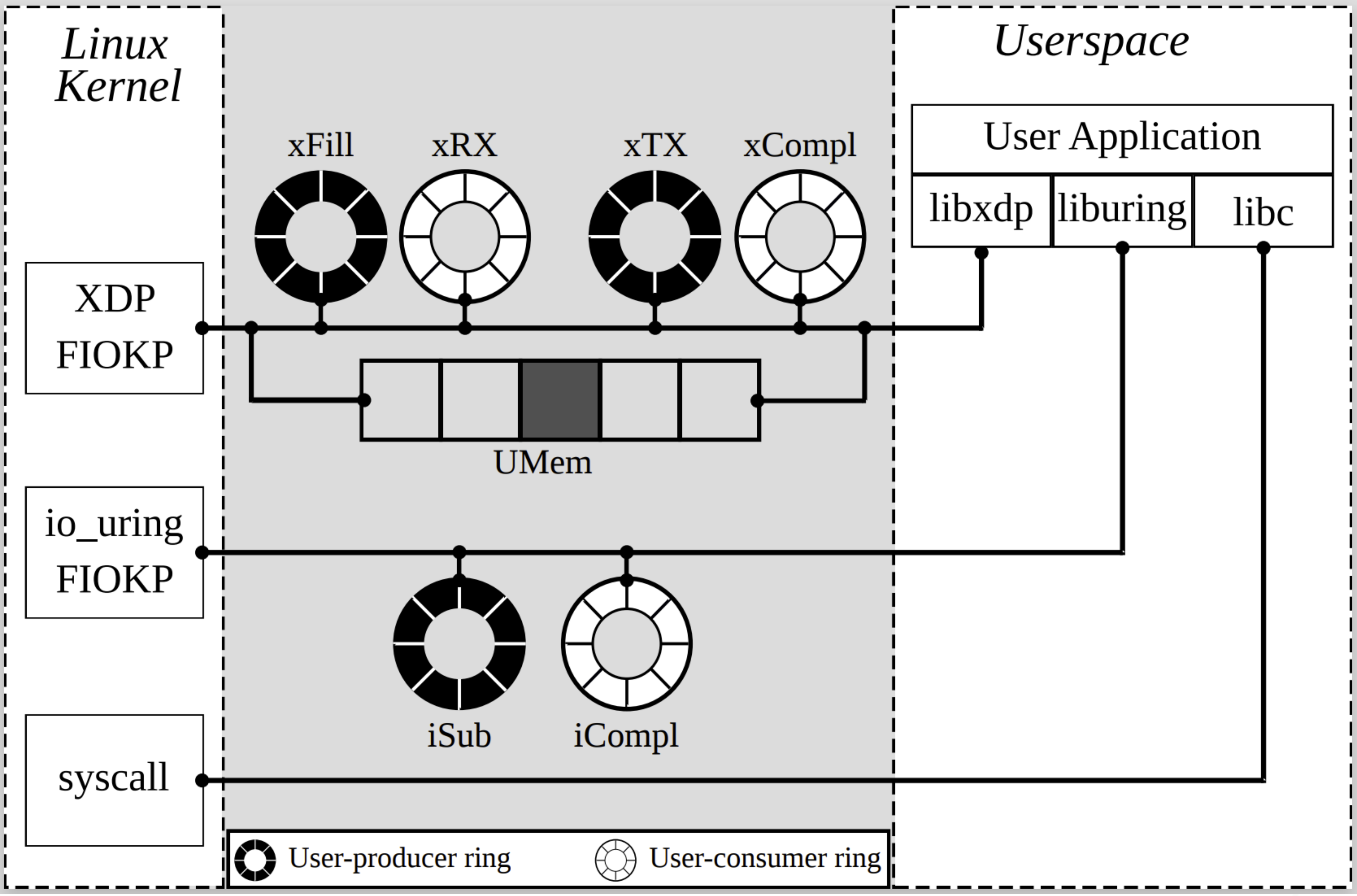
Fast I

- Recen
  - eXp
  - io_u
- These ations.
- They e th syscal



Fast I

- Recen
  - eXp
  - io_u
- These ations.
- They e th syscal



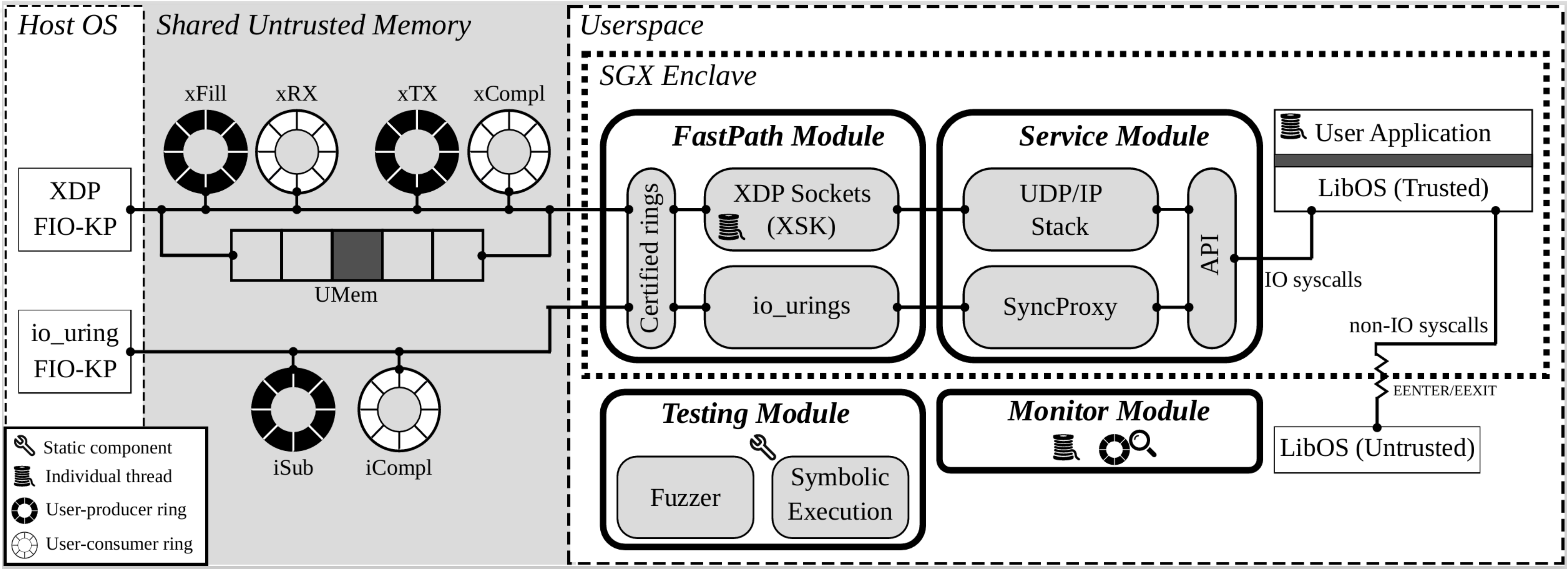
RAKIS - Challenges

1. FIOKPs and their userspace libraries assume a trusted OS.
   - This assumption does not extend to enclave programs.

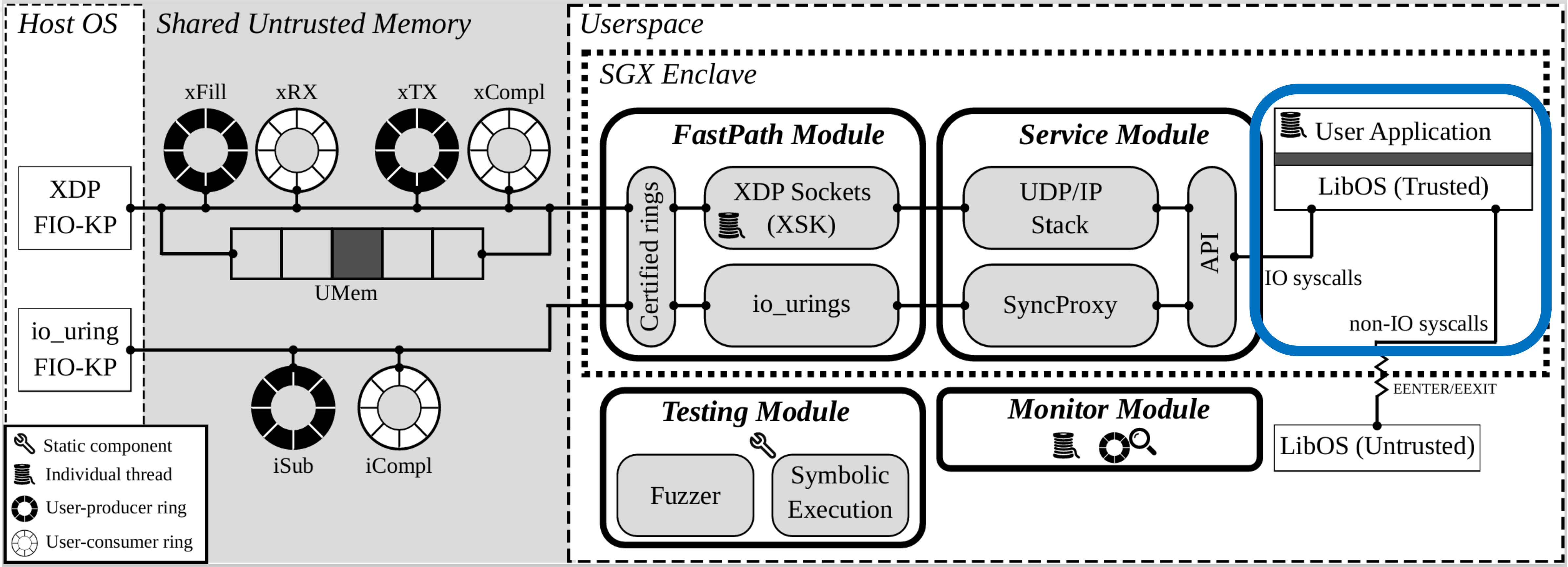
2. FIOKPs have in-compatible IO interfaces to regular IO syscalls.
   - This necessitates modifications to enclave programs.

3. FIOKPs services do not match enclave program expectations.
   - XDP operate on layer-2 data-frames only.
   - io_uring only handles asynchronous syscalls.

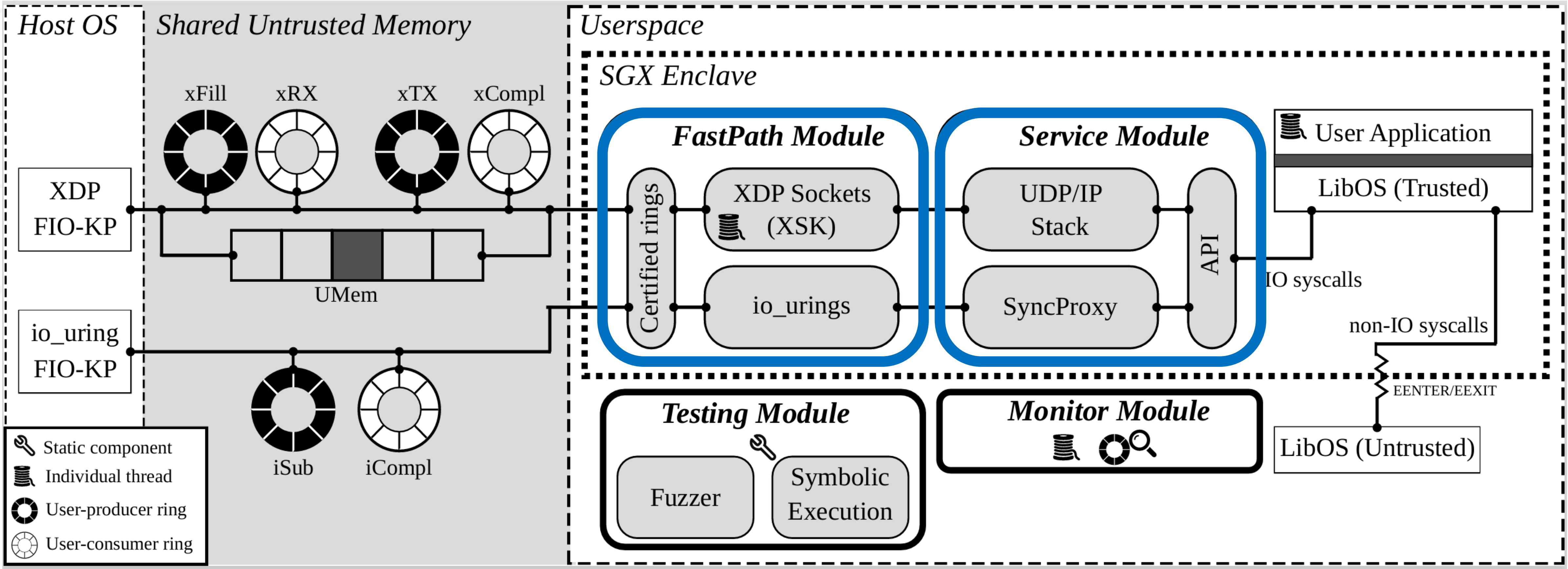
RAKIS: Design



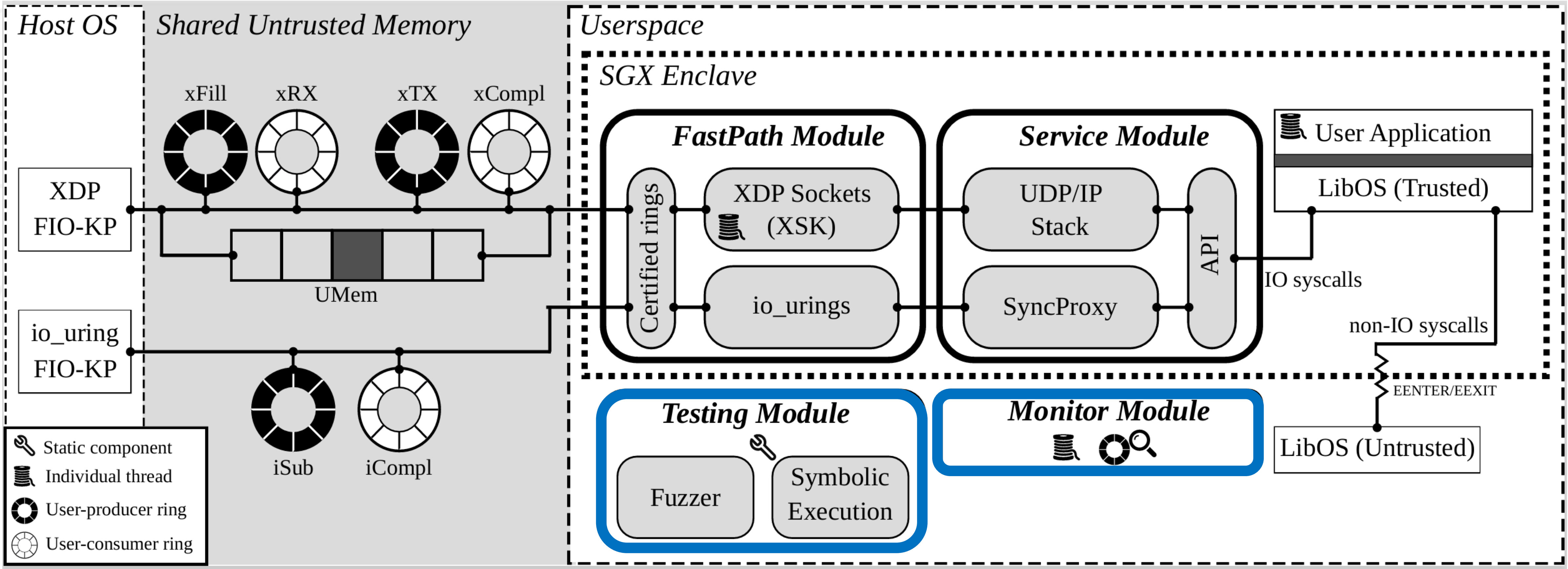
RAKIS: Design - Enclave Modules



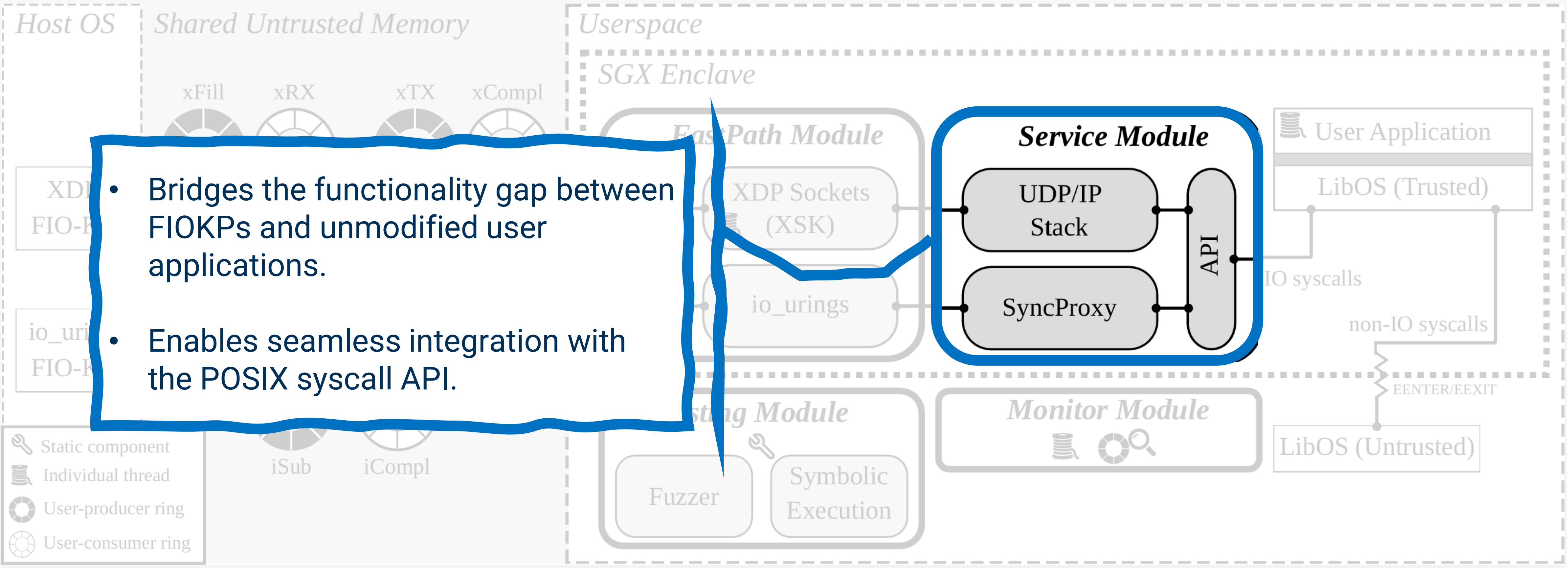
RAKIS: Design - Enclave Modules



RAKIS: Design - Userspace Modules

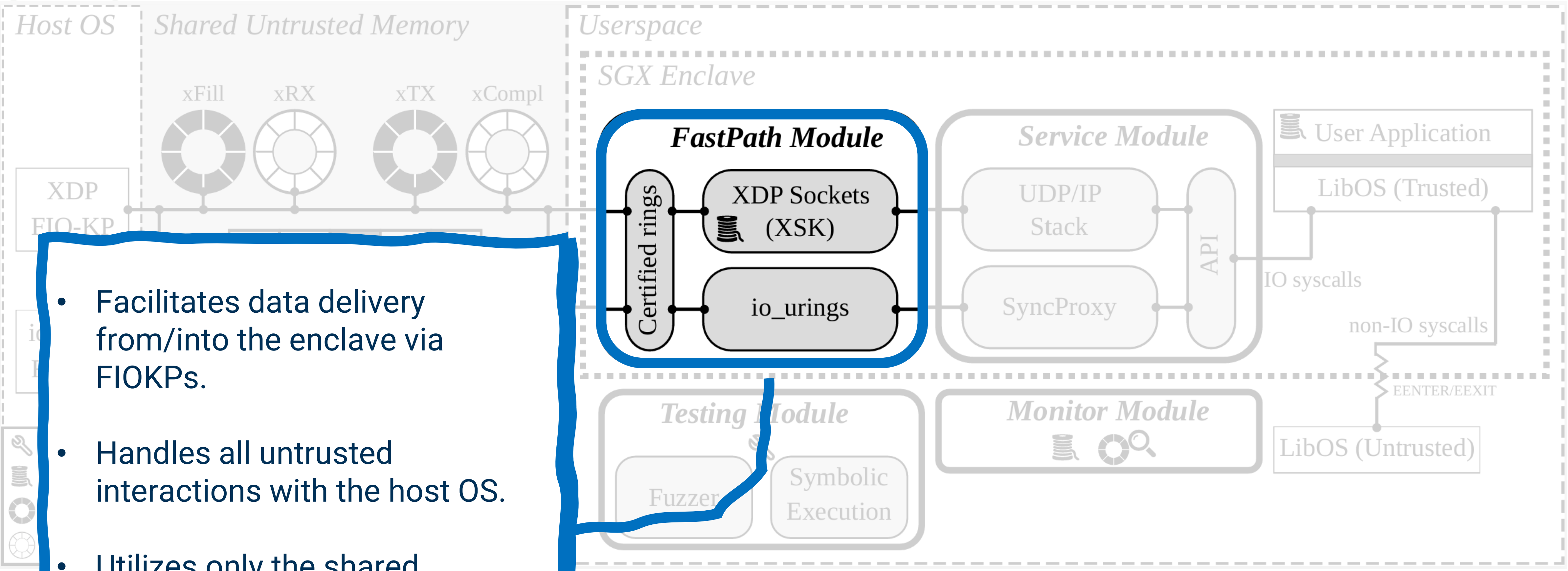


RAKIS: Design - Service Module



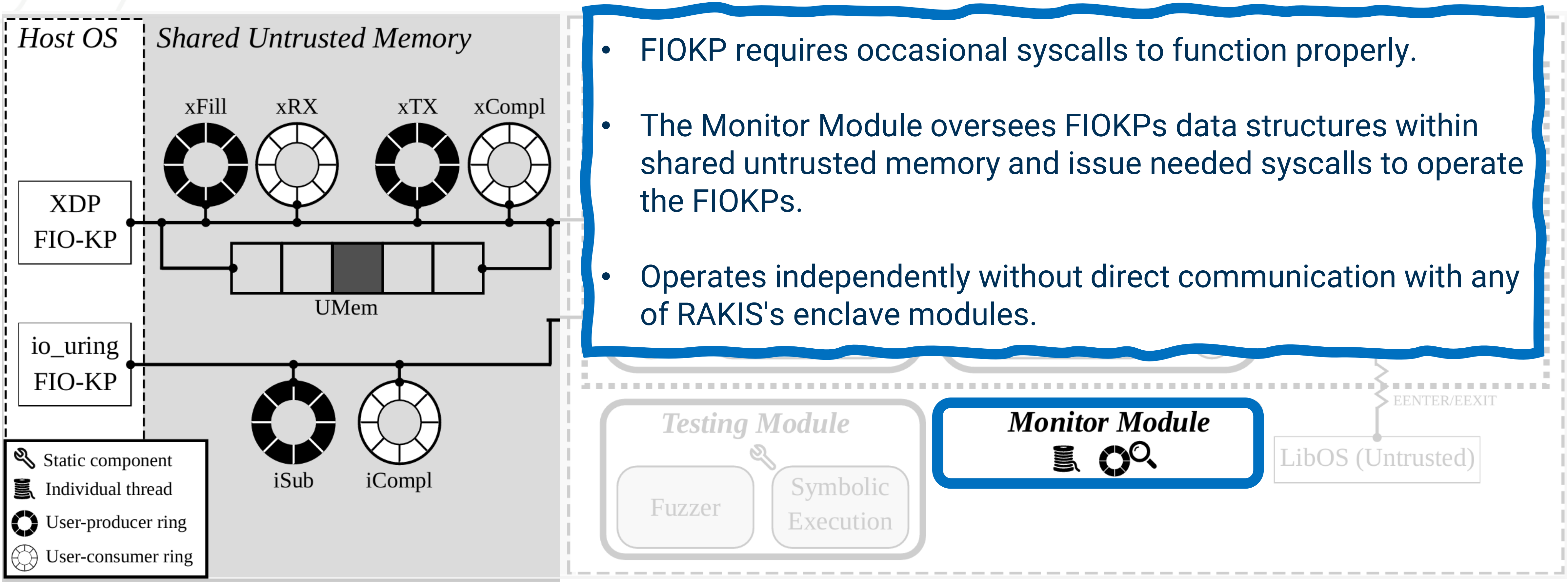
- Bridges the functionality gap between FIOKPs and unmodified user applications.
- Enables seamless integration with the POSIX syscall API.

RAKIS: Design - FastPath Module



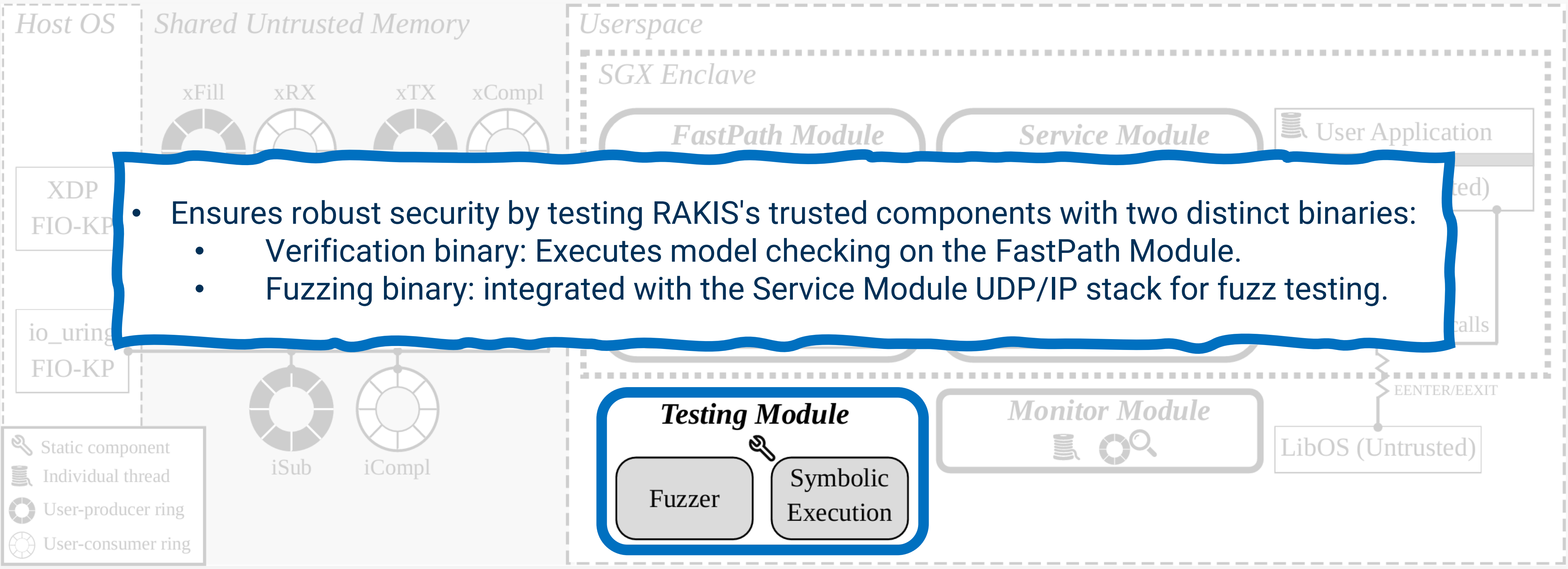
- Facilitates data delivery from/into the enclave via FIOKPs.
- Handles all untrusted interactions with the host OS.
- Utilizes only the shared untrusted memory without requiring any enclave exits.

RAKIS: Design - Monitor Module



- FIOKP requires occasional syscalls to function properly.
- The Monitor Module oversees FIOKPs data structures within shared untrusted memory and issue needed syscalls to operate the FIOKPs.
- Operates independently without direct communication with any of RAKIS's enclave modules.

RAKIS: Design - FastPath Module



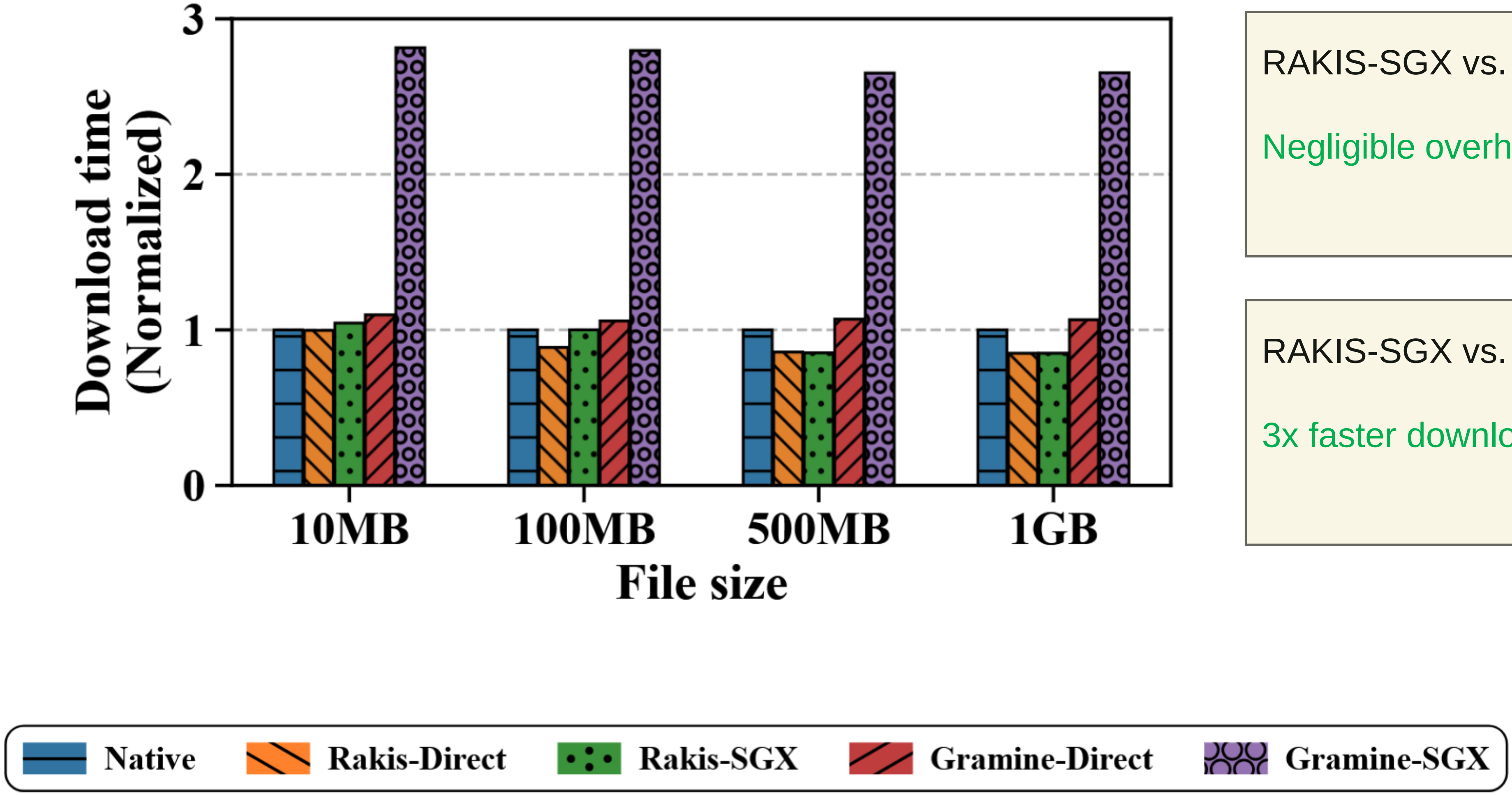
- Ensures robust security by testing RAKIS's trusted components with two distinct binaries:
  - Verification binary: Executes model checking on the FastPath Module.
  - Fuzzing binary: integrated with the Service Module UDP/IP stack for fuzz testing.

RAKIS: Performance Evaluation

- Five runtime environments:
  1. Native: host OS(program).
  2. Gramine-Direct: host OS(Gramine(program)).
  3. RAKIS-Direct: host OS(RAKIS(program)).
  4. Gramine-SGX: host OS(SGX_Enclave(Gramine(program)).
  5. RAKIS-SGX: host OS(SGX_Enclave(RAKIS(program)).

- Six workloads:
  1. iperf (UDP IO).
  2. Curl (UDP IO).
  3. Memcached (UDP IO).
  4. fstime (File IO).
  5. Redis (TCP IO).
  6. MCrypt (File IO).

RAKIS: Performance Evaluation (Curl – UDP IO over XDP)



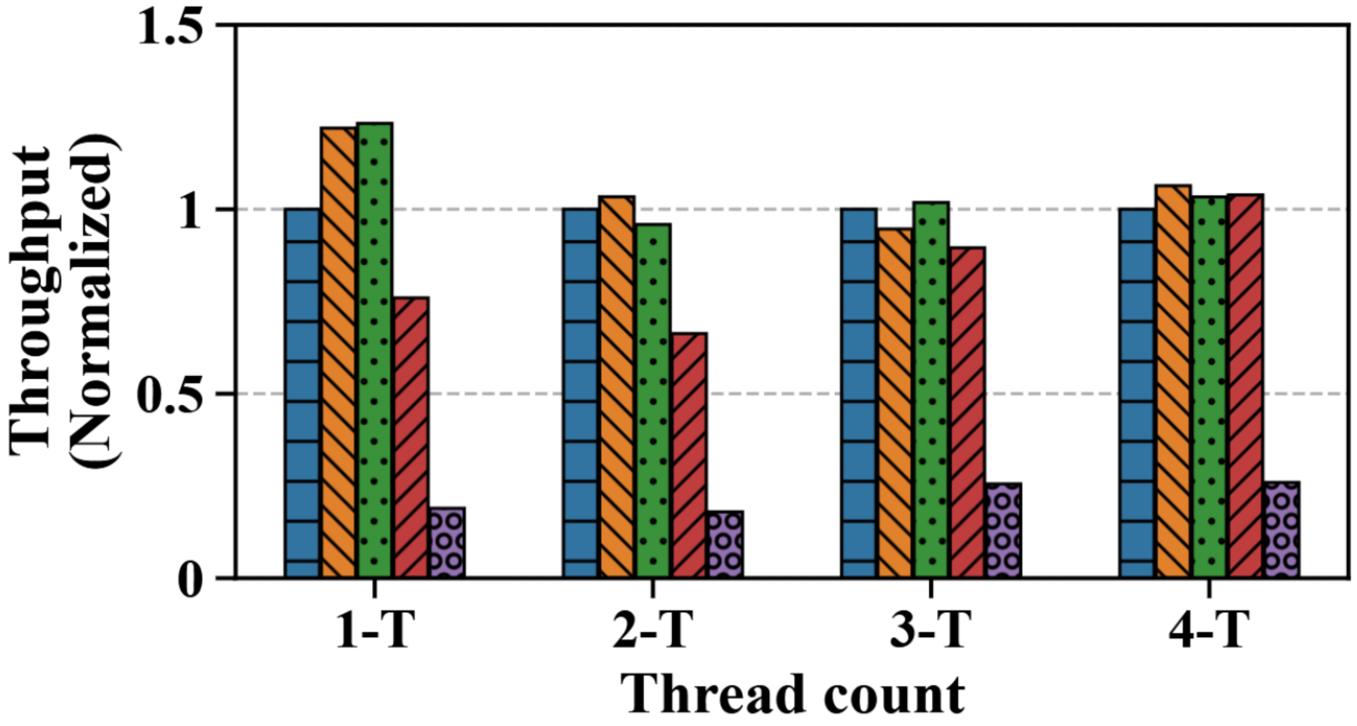
RAKIS-SGX vs. NATIVE:

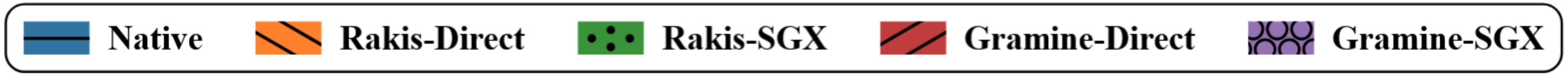
Negligible overhead.

RAKIS-SGX vs. Gramine-SGX:

3x faster download times.

RAKIS: Performance Evaluation (memcached – UDP IO over XDP)





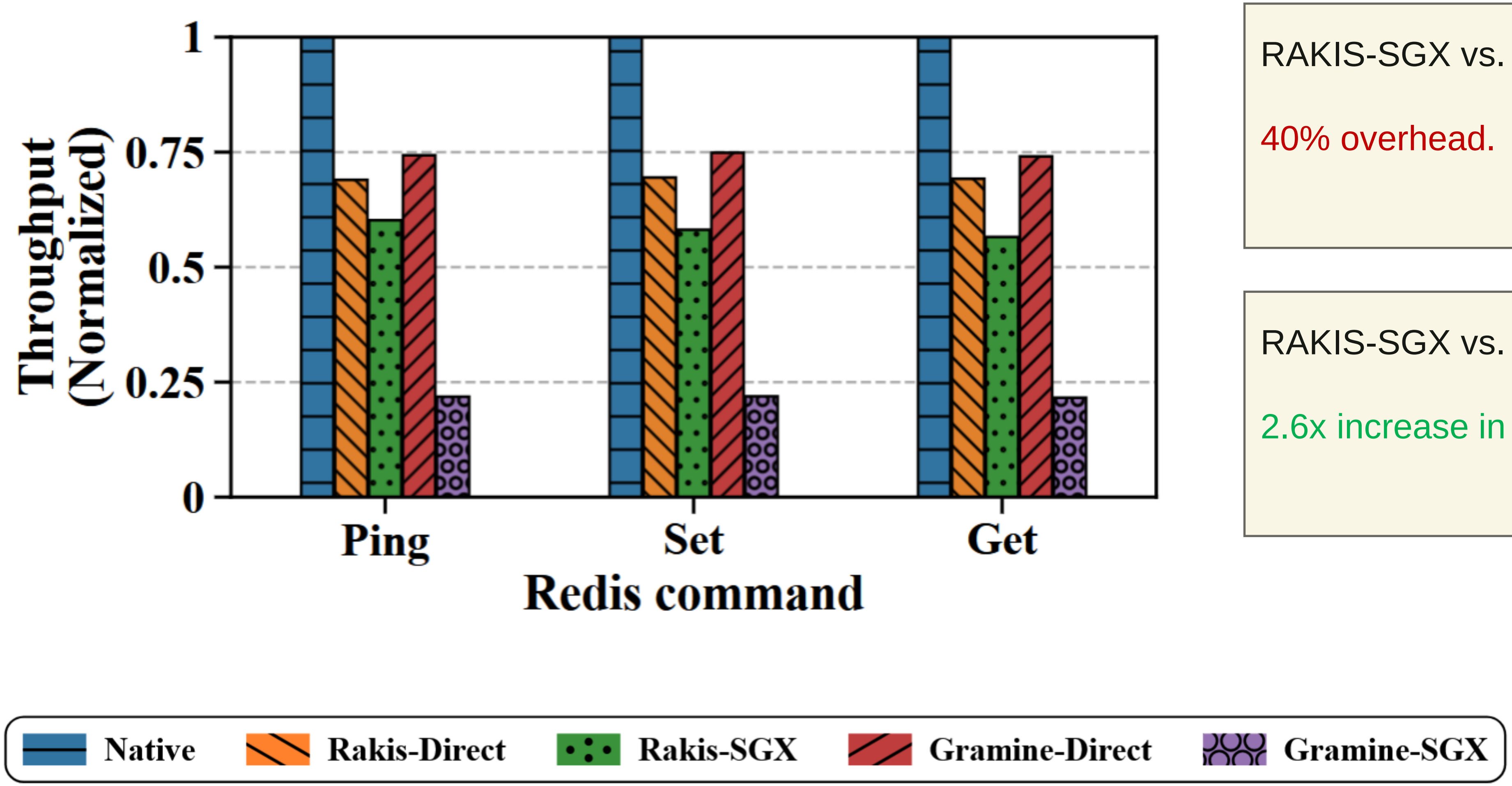
RAKIS-SGX vs. NATIVE:

Negligible overhead.

RAKIS-SGX vs. Gramine-SGX:

4.6x increase in throughput.

RAKIS: Performance Evaluation (Redis – TCP IO over io_uring)



RAKIS-SGX vs. NATIVE:

40% overhead.

RAKIS-SGX vs. Gramine-SGX:

2.6x increase in throughput.

RAKIS: Simpler, Lighter, and More Efficient

- RAKIS does not require any special hardware.
  - Only requires new kernels where XDP and io_uring is supported.
- RAKIS have a small footprint.
  - Less than 8K LoC.
  - Tested with Symbolic execution and fuzzing.
- Tailored for user workload.
  - Does not necessitate heavy OS features.
  - Tunable CPU cores and memory footprint.

Conclusion

- RAKIS securely enables fast IO primitves inside SGX enclaves.
  - Runs unmodified user programs.
  - Small & extensively tested TCB.
  - Easy to deploy.
  - Resource efficient.
  - Achieves an average improvement of 2.8x compared to Gramine-SGX across all workloads.

- Open source:
  - https://github.com/sslab-gatech/RAKIS

Artifacts Evaluated acm Functional V1.1

Artifacts Available acm V1.1

Results Reproduced acm V1.1

Q&A

- Mansour Alharthi.
- 6th year Ph.D student @ GeorgiaTech
- Email: mansourah@gatech.edu