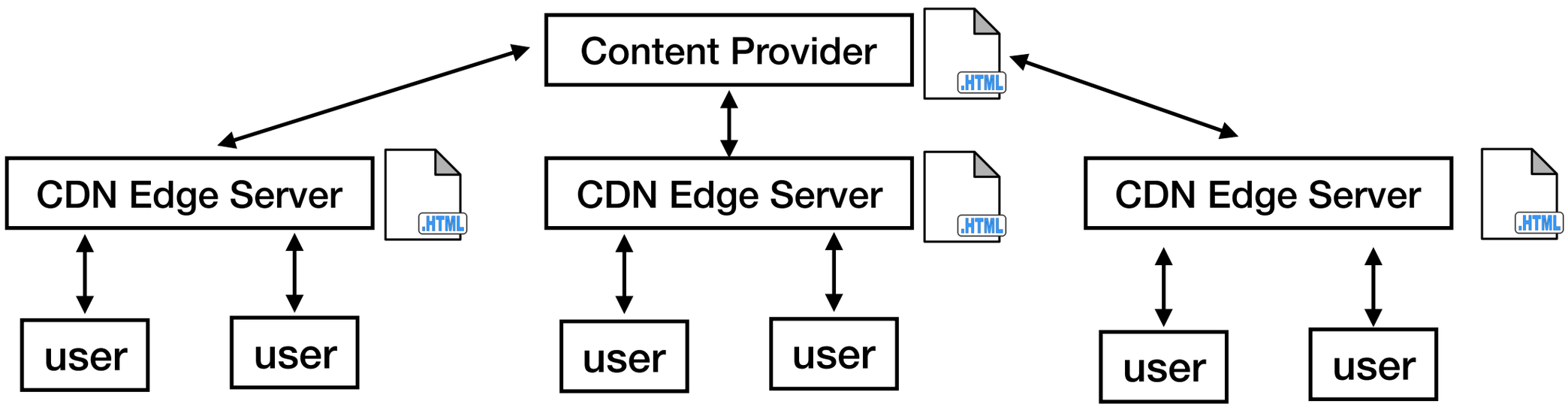
SecureCDN: Providing End-to-End Security in Content Delivery Networks

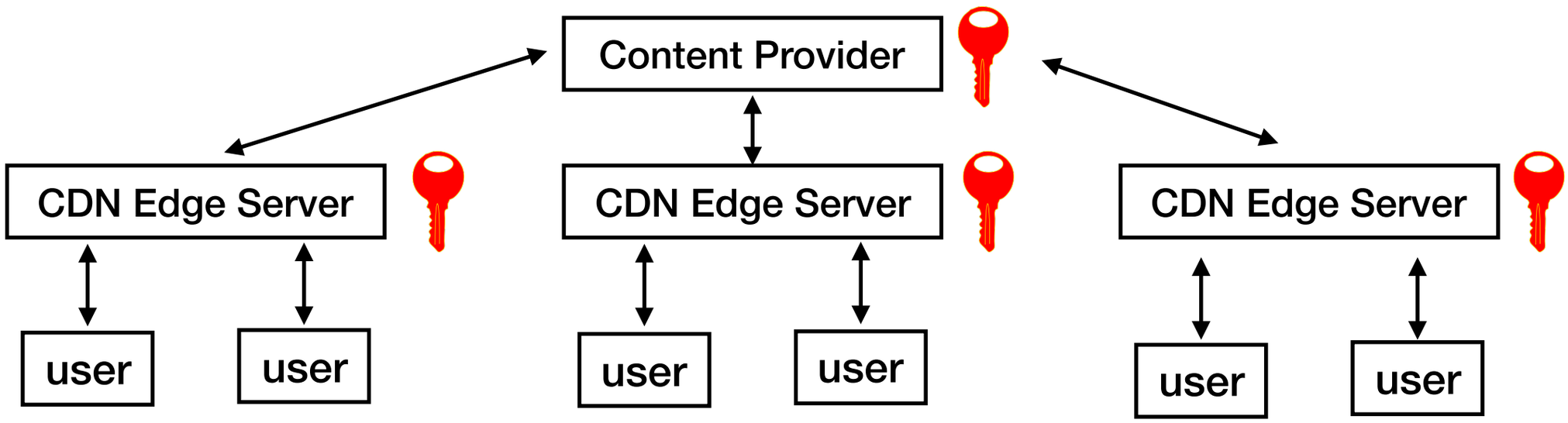
Stephen Herwig
University of Maryland, College Park

Content Delivery Networks



Performance
Scalability
Security

CDNs and HTTPS

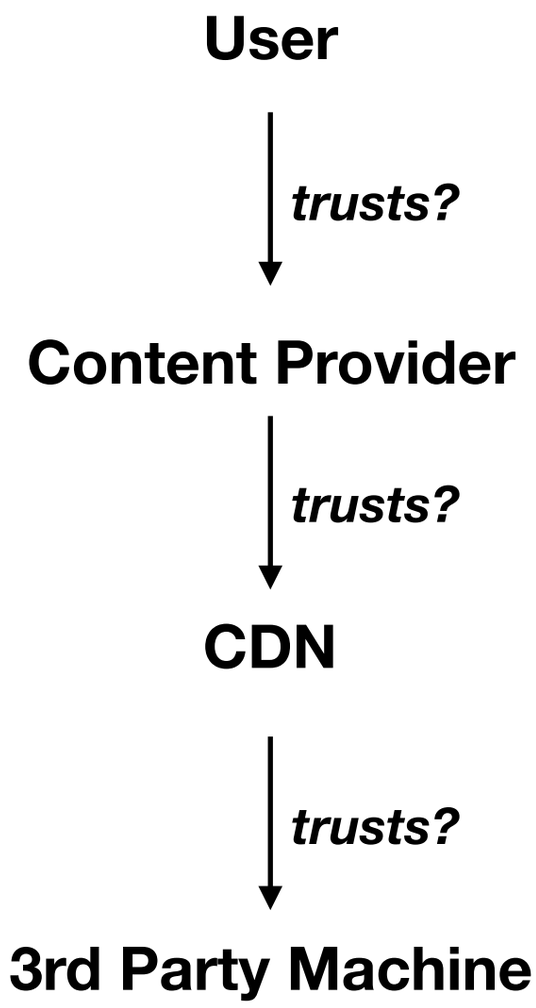


Performance
Scalability
Security?

Liang, et al., When HTTPS meets CDN: A Case of Authentication in Delegated Service. IEEE S&P, ’14

Cangialosi et al., Measurement Analysis of Private Key Sharing in the HTTPS Ecosystem, CCS, ’16

Problem: Strained Trust Model



<u>Additional Complications</u>:

- Future legislation compelling intermediary liability
- National Security Letters for data request

Cast as “Delegation" Problem

Threat Model

Null

Approach

- **X. 509 extensions expressing “*A authorizes B to perform an action.*”**

Tuck et al., Internet X.509 Public Key Infrastructure Proxy Certificate Profile. (draft-ietf-pkix-proxy-03), 2002

Cooper et. al, RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, Section 4.2.1.10 “Name Constraints”, 2008

- **DANE extensions “…”**

Liang, et al., When HTTPS meets CDN: A Case of Authentication in Delegated Service. IEEE S&P, ’14.

Cast as “Coupling of Auth/Integrity with Distribution” Problem

Threat Model

CDN may modify content and/or try to impersonate Content Provider
“Trust but verify”

Approach

- **Application layer: User obtains signed manifest from Content Provider.**

Levy et al., Stickler Defending Against Malicious CDNs in an Unmodified Browser, IEEE S&P ’16.

- **Transport layer: Content Provider and CDN cooperatively create TLS stream.**

Lesnieski-Lass and Kasshoek, SSL splitting: securely serving data from untrusted caches, USENIX Security ’03.

Nick Sullivan, Keyless SSL: The Nitty Gritty Details, https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-details/, 2014

Cast as “Secure Remote Computation” Problem

Definition

Secure remote computation is the problem of executing software on a remote computer **owned and maintained by an untrusted party**, with some integrity and confidentiality guarantees**.**

Motivates revised CDN trust model:

Can the Content Provider reduce the adversarial power of the CDN to that of a traditional on-path HTTPS adversary?

Intel Secure Guard Extensions (SGX)

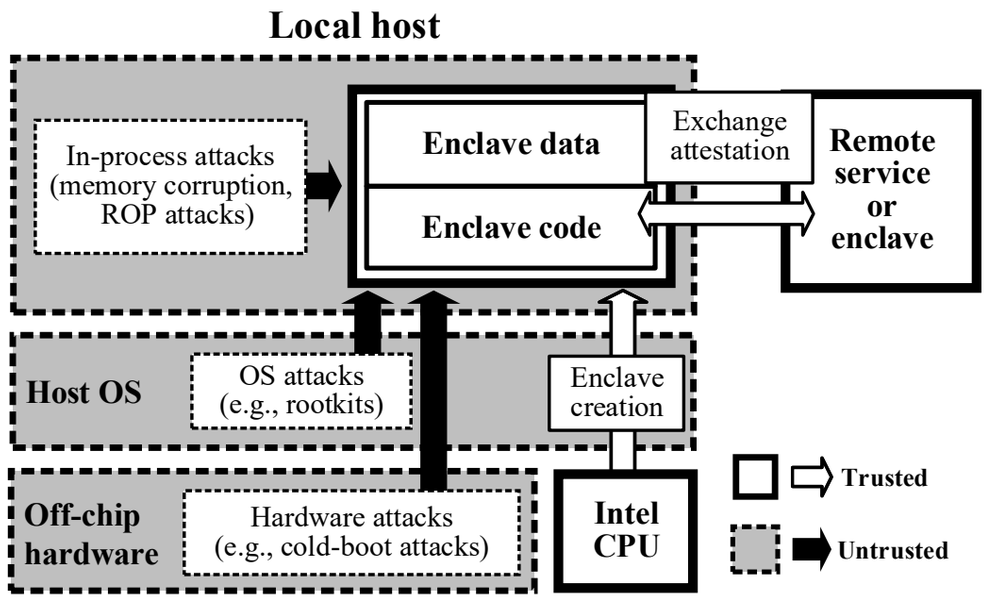


Figure 1: The threat model of SGX. SGX protects applications from three types of attacks: in-process attacks from outside of the enclave, attacks from OS or hypervisor, and attacks from off-chip hardware.

Threat Model

Enclave code author need only trust the CPU
Untrusted System can always deny service

Limitations

Total enclave memory restricted to 128 MB
Enclave cannot explicitly share memory pages with other processes
An RPC out of the enclave is 8,200 - 17,000 cycles (vs. 150 for a typical syscall)

Approach: Minimal Code in Enclave

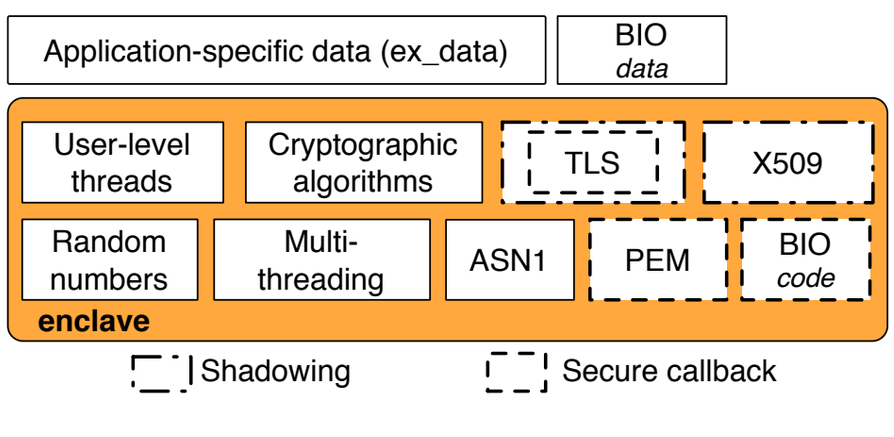


Figure 1: TaLoS TLS implementation

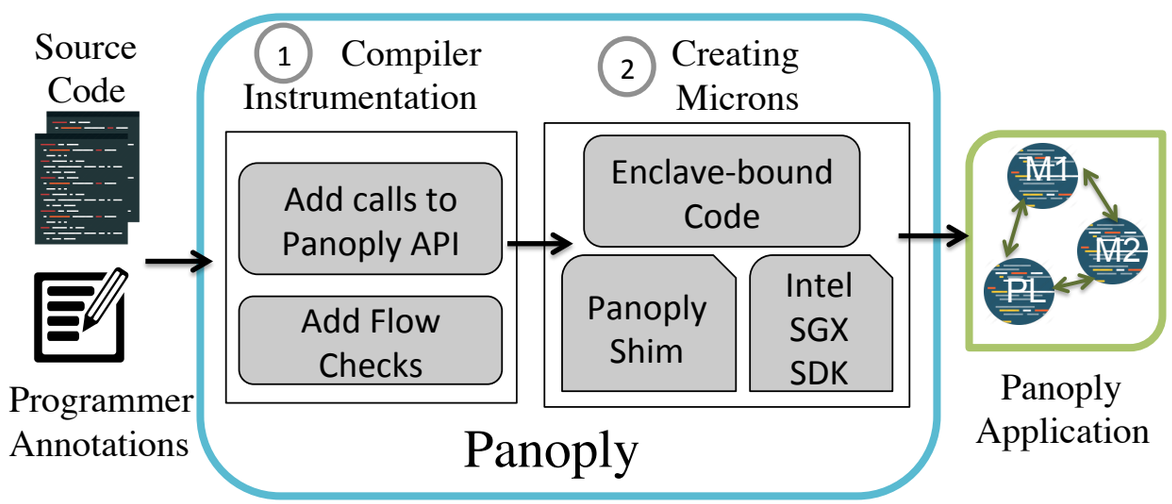


Fig. 4. System Overview. PANOPLY takes in the original program and the partitioning scheme as input. It first divides the application into enclaves and then enforces inter-micron flow integrity, to produce PANOPLY application.

Aublin et. al, TaLoS: Secure and Transparent TLS Termination inside SGX Enclave, Technical Report, '17.

Shinde et. al, PANOPLY: Low-TCB Linux Applications with SGX Enclaves, NDSS '17.

Approach: LibOS in Enclave

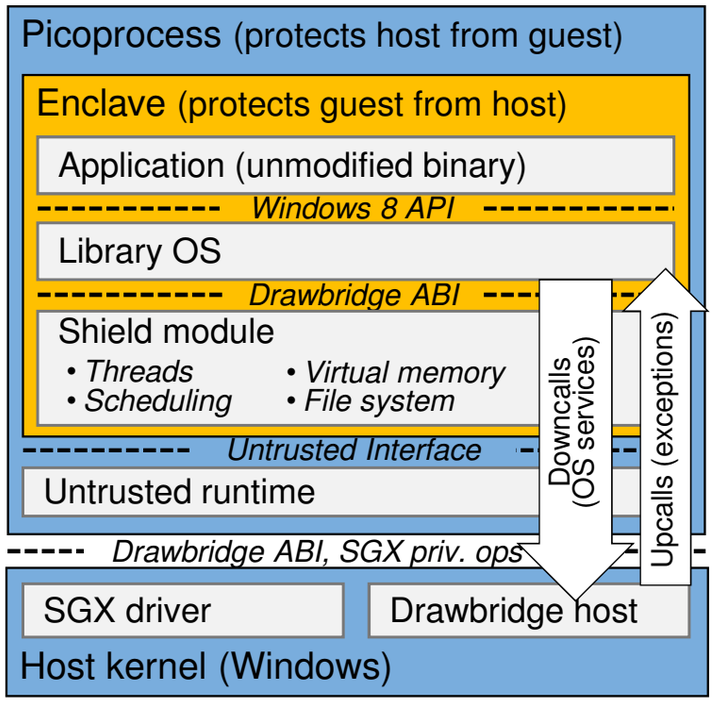


Figure 2: Haven components and interfaces

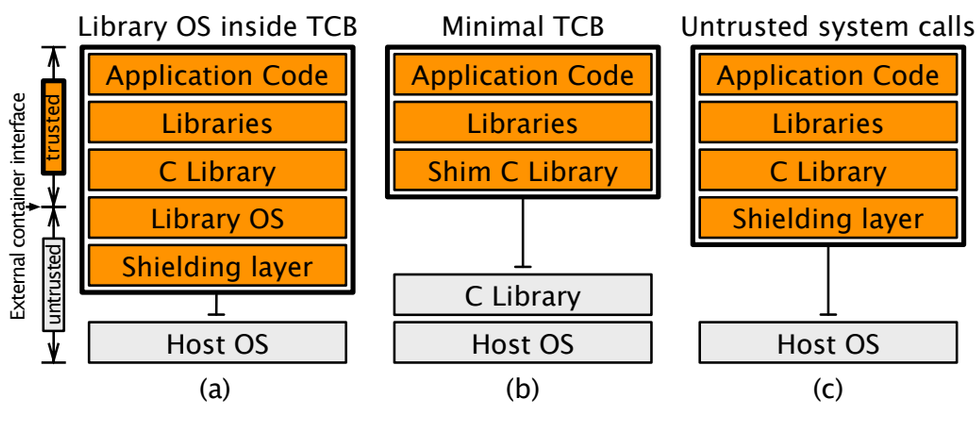


Figure 1: Alternative secure container designs

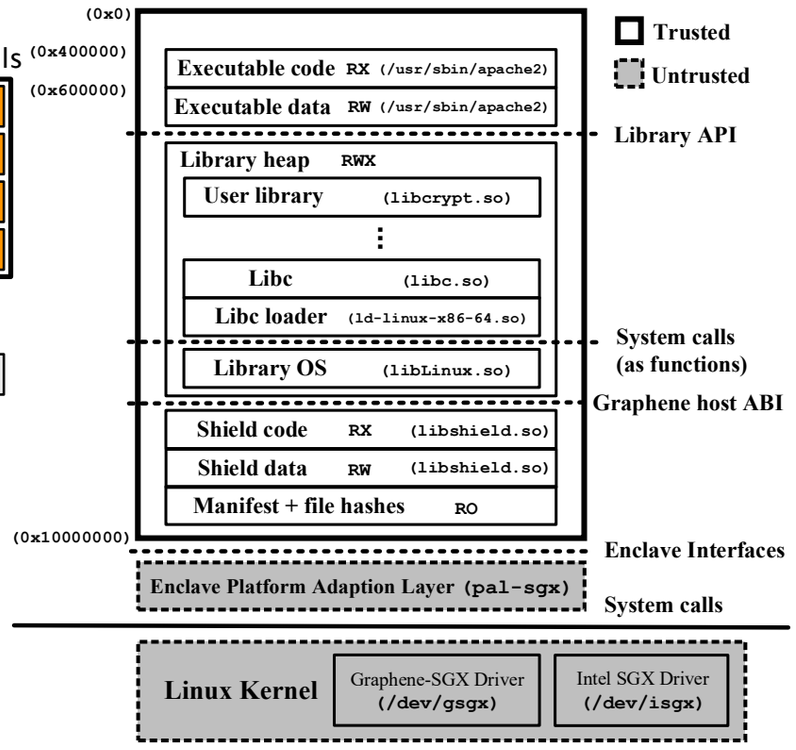


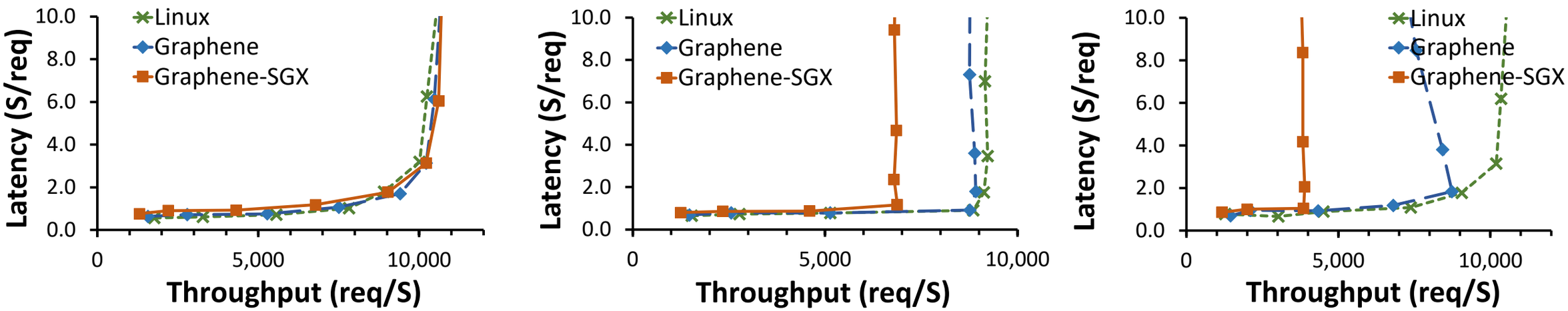
Figure 3: The Graphene-SGX architecture. The executable is position-dependent. The enclave includes an OS shield, a library OS, libc, and other user binaries.

Baumann et al., Shielding Applications from an Untrusted Cloud with Haven, OSDI '14

Amautov et al., SCONE: Secure Linux Containers with Intel SGX, OSDI '16

Tsai et al., Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX, USENIX ATC '17

SGX LibOS Performance



(a) Lighttpd (25 threads) **(b) Apache (5 processes)** **(c) NGINX (event-driven)**

Figure 5: Throughput versus latency of web server workloads, including Lighttpd, Apache, and NGINX, on native Linux, Graphene, and Graphene-SGX. We use an ApacheBench client to gradually increase load, and plot throughput versus latency at each point. Lower and further right is better.

Latency is 12-35% more than native
For Apache, peak throughput is 75% of native
For NGINX, peak throughput is 40% of native

Current SGX LibOS Shortcomings

Multiprocess Abstractions

Haven & SCONE:

limited to a single process

Graphene-SGX:

Implements fork as process migration. Limited support for POSIX IPC / shared memory

Filesystems

Haven:

Encrypted virtual disk image formatted as FAT filesystem

SCONE:

For security guarantees, a union fs: host is read-only; writes copy file to in-memory fs

Graphene-SGX:

For security guarantees, host fs is read-only

Time

All: To prevent Iago attacks, need a trusted source of time

Availability

Haven & SCONE:

Closed source

Graphene-SGX:

Open-sourced (https://github.com/oscarlab/graphene)

Remaining Threats

An untrusted may still observe:

- **Executables that are run and the libraries that they load**

- **Shape of the process trees, IPC relationships, resource usage**

- **Access patterns to the libOS's filesystem**

  Use a filesystem with ORAM properties?

  Ahmad et al., OBLIVIATE: A Data Oblivious File System for Intel SGX, NDSS '18

- **Fingerprints of web requests (e.g., object sizes)**

- **Linkability of client requests**

- **Socket metadata and network traffic patterns**

  Move the network stack into the libOS; incorporate VPN/ Tor into this stack?

Larger Goal: Oblivious Host

Although we framed the problem as a Secure CDN, are we really aiming for an ***oblivious host*** — a host that is “unaware” of the processes it is running?