Chancel: efficient multi-client isolation under adversarial programs

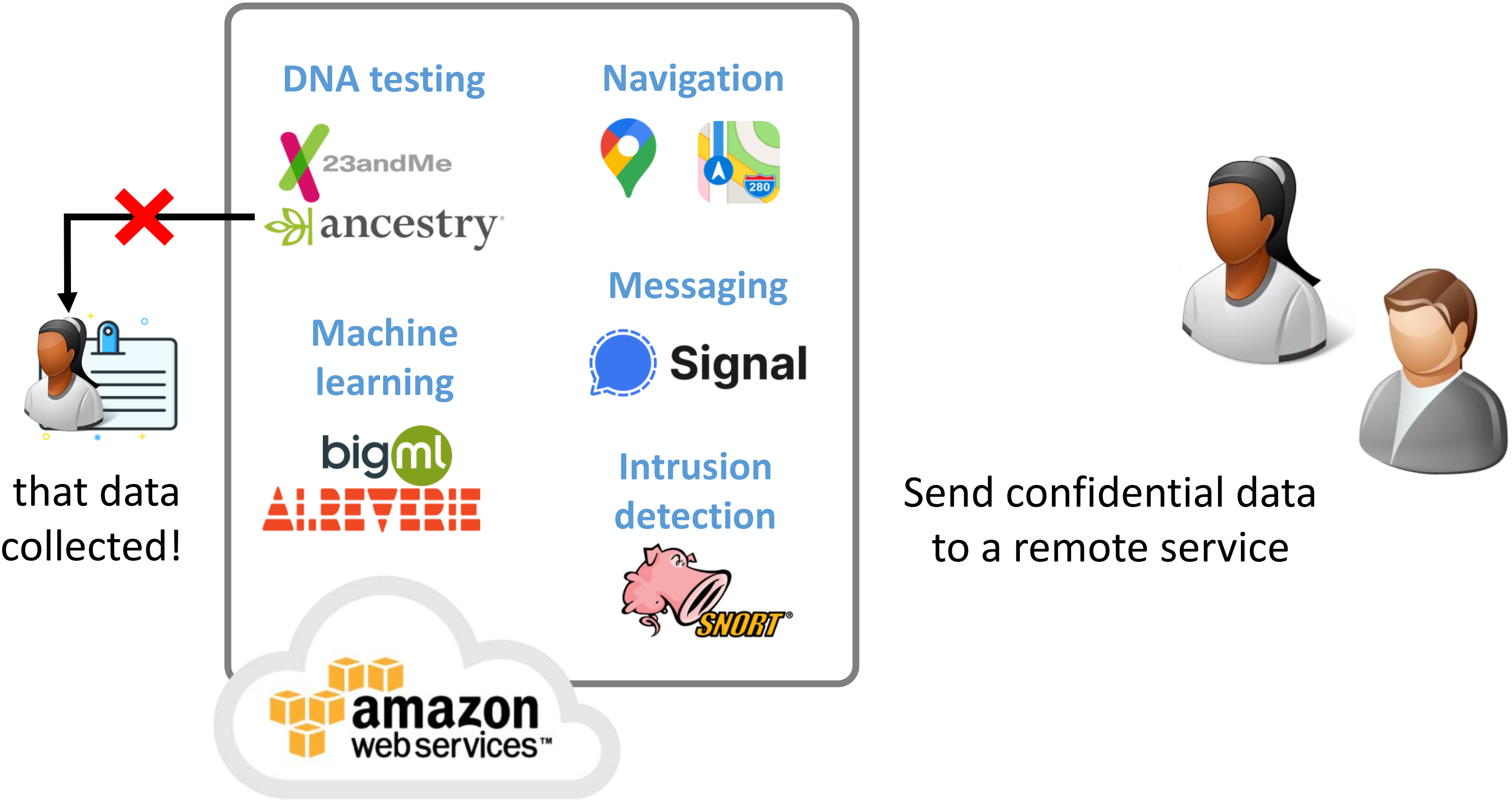
Adil Ahmad, Juhee Kim, Jaebaek Seo,
Insik Shin, Pedro Fonseca, and Byoungyoung Lee

PURDUE UNIVERSITY

SEOUL NATIONAL UNIVERSITY

KAIST

Data security in sensitive remote services



DNA testing

Navigation

Messaging

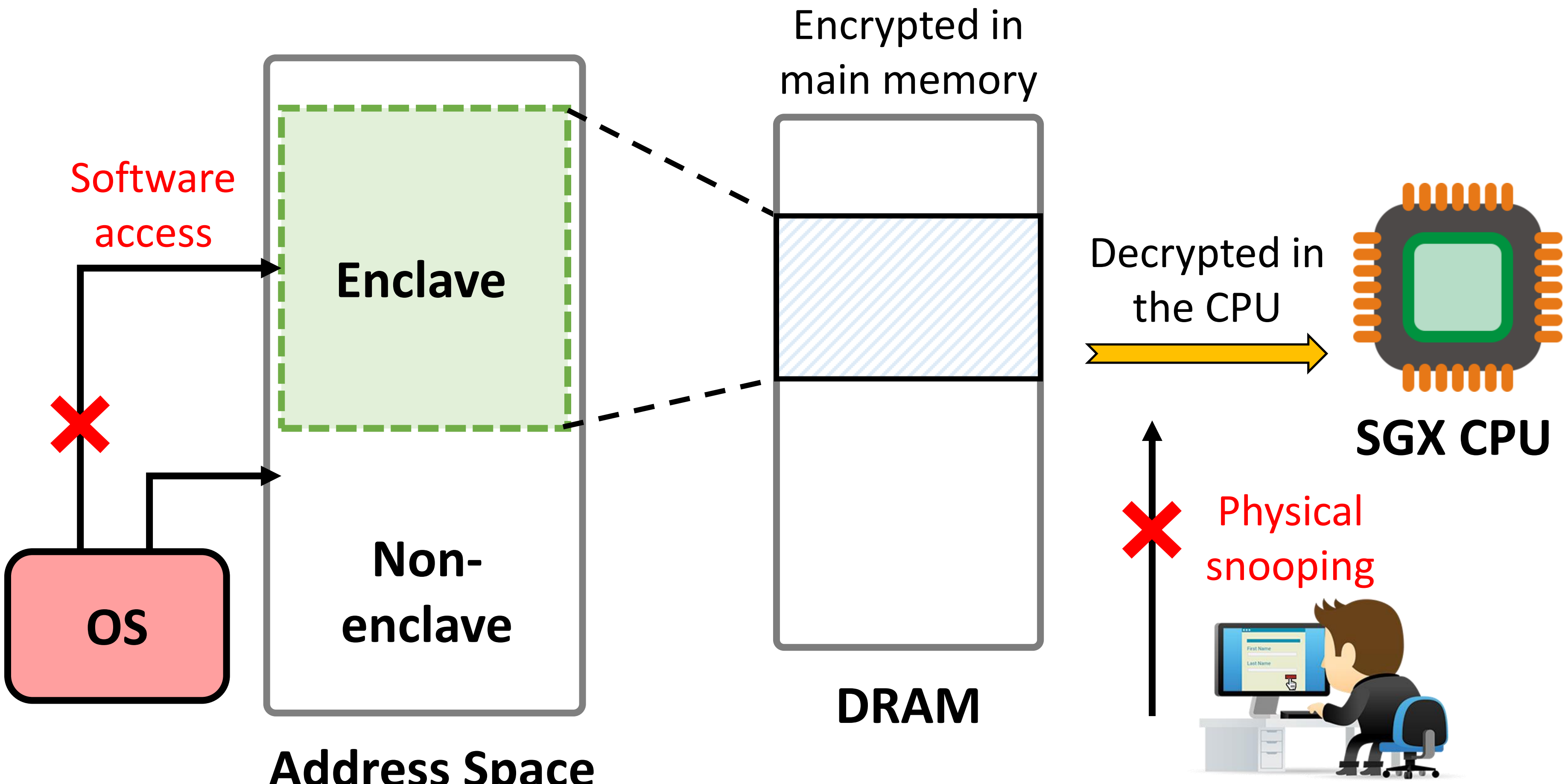
Machine learning

Intrusion detection

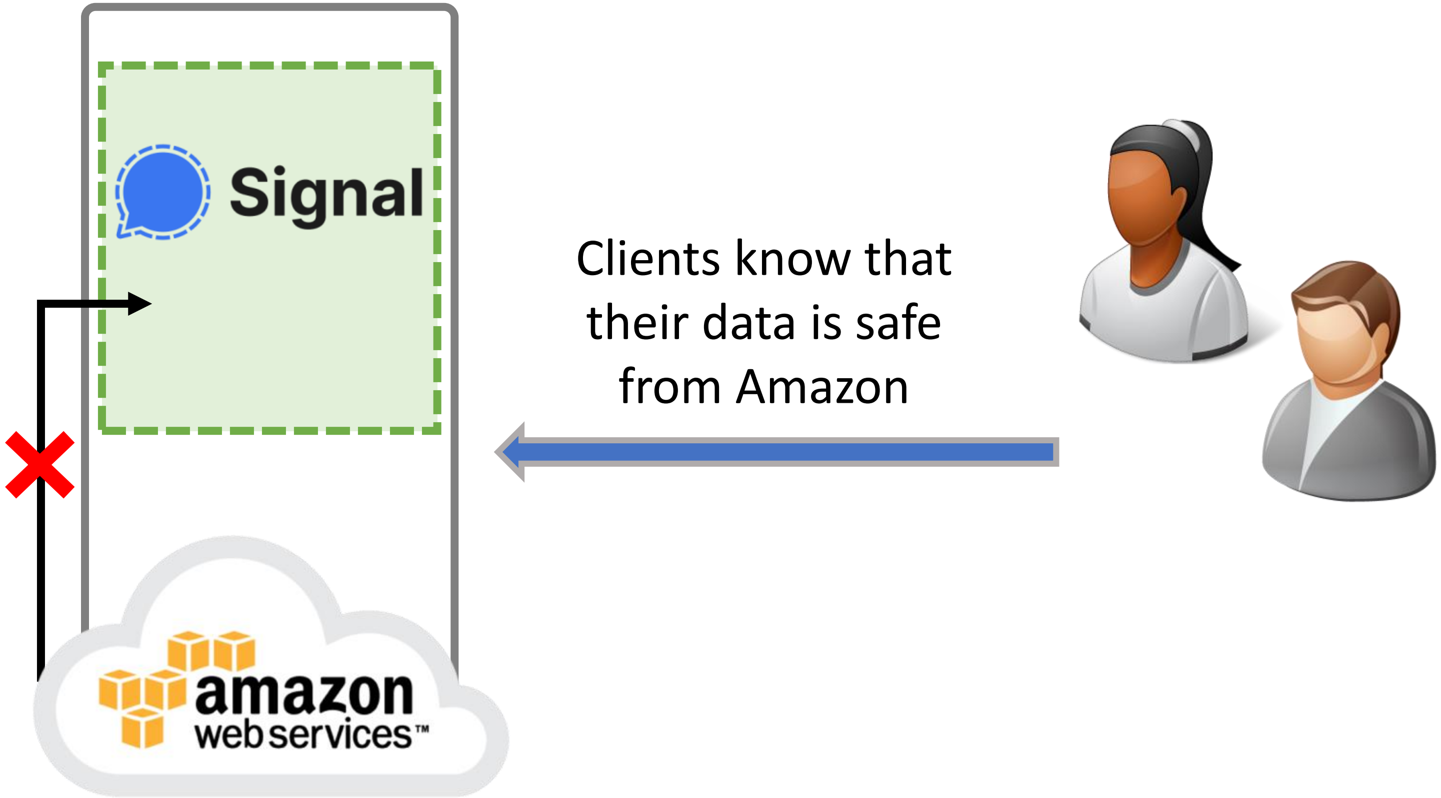
Guarantee that data cannot be collected!

Send confidential data to a remote service

SGX partially secures remote data



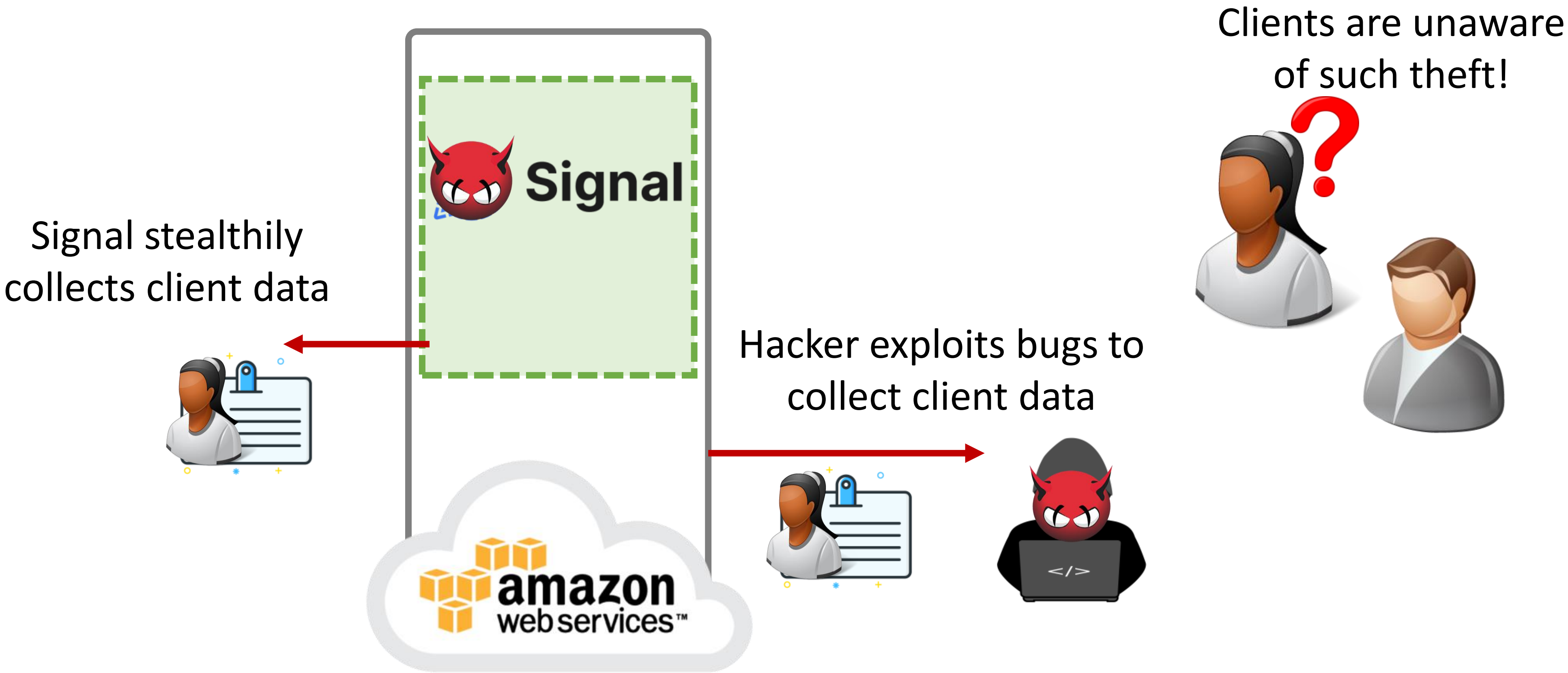
SGX secures remote data from clouds



Signal uses SGX; Amazon cannot access Signal's contents

Clients know that their data is safe from Amazon

SGX does not secure data from untrusted code



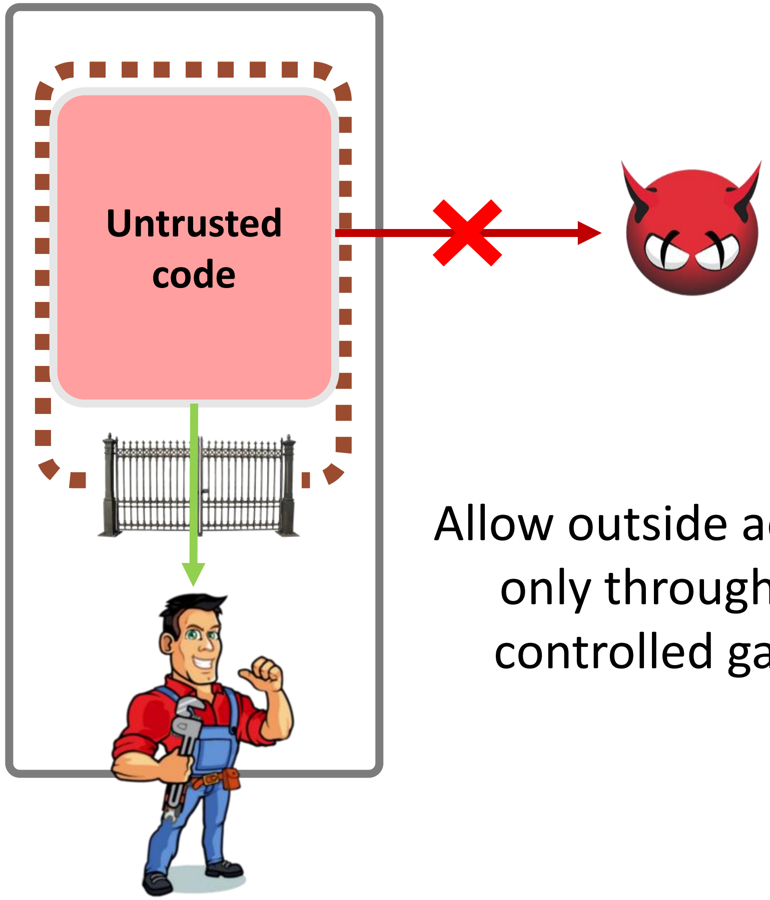
Signal stealthily collects client data

Hacker exploits bugs to collect client data

Clients are unaware of such theft!

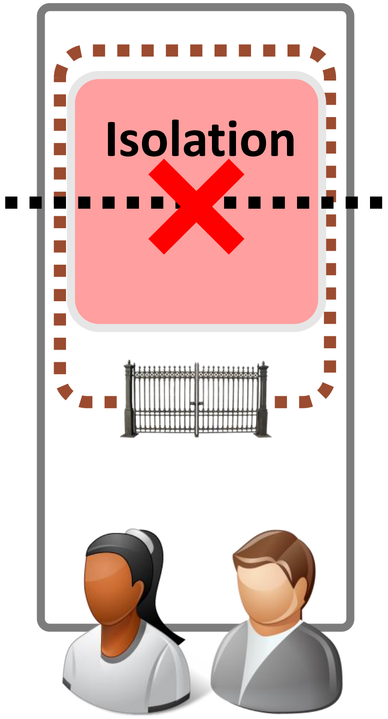
Software Fault Isolation restricts untrusted code

Create a brick wall around untrusted code



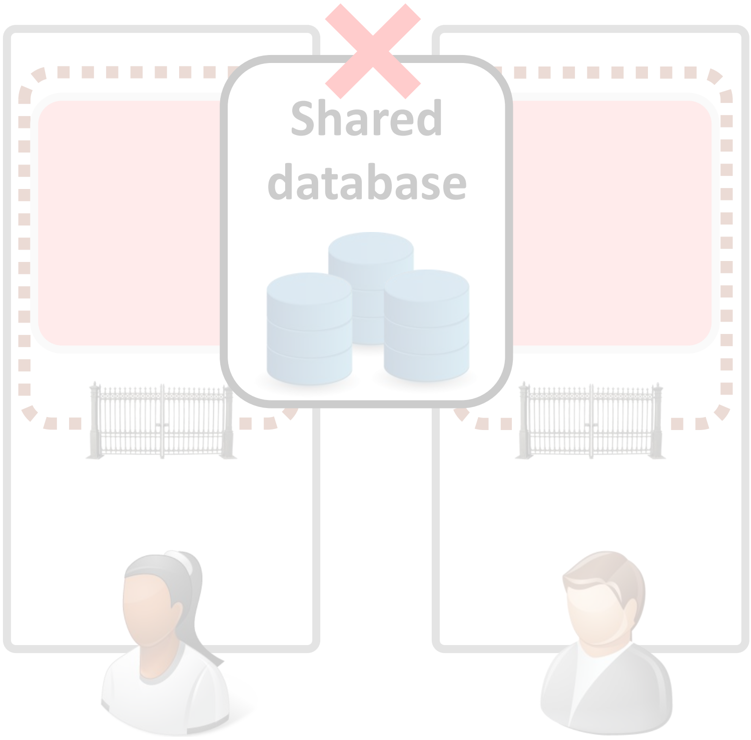
Allow outside access only through a controlled gate

Native Client SFI requires multiple processes

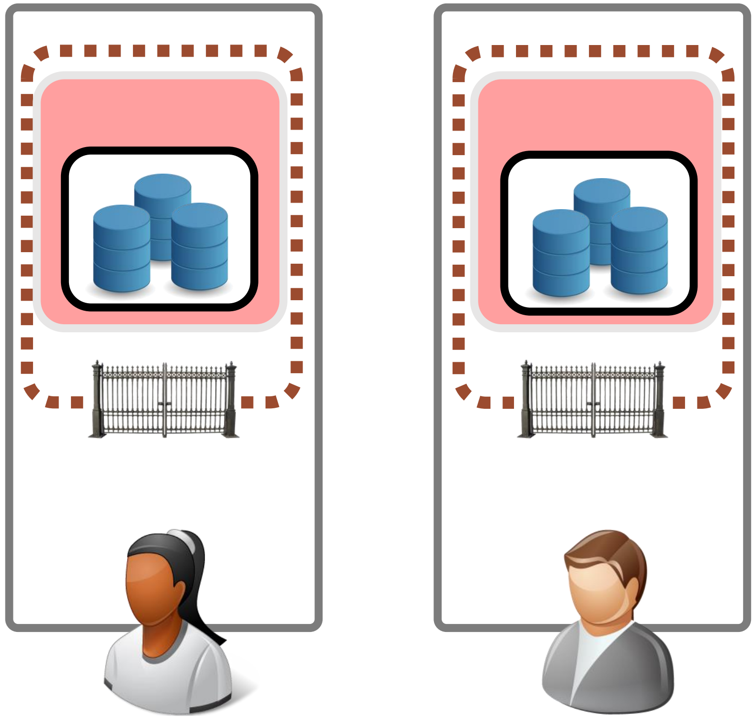


Cannot serve multiple clients in a process

Multiple processes consume a lot of memory

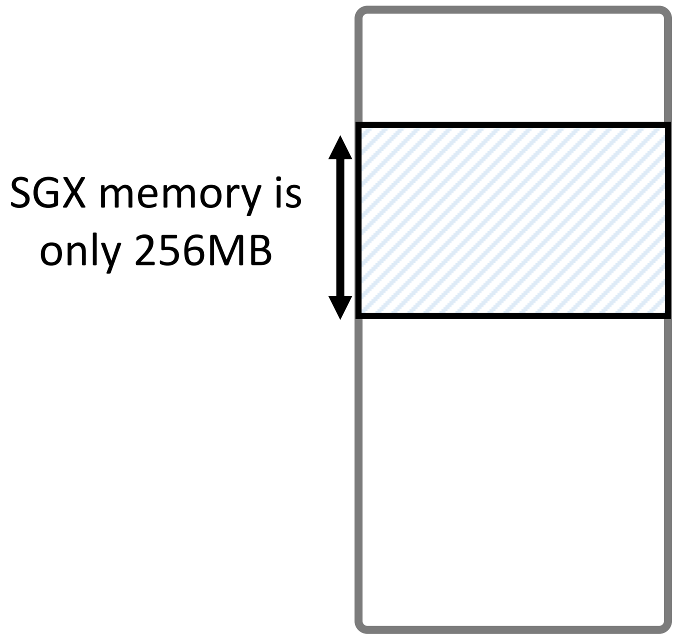


Lack efficient and secure inter-process memory sharing



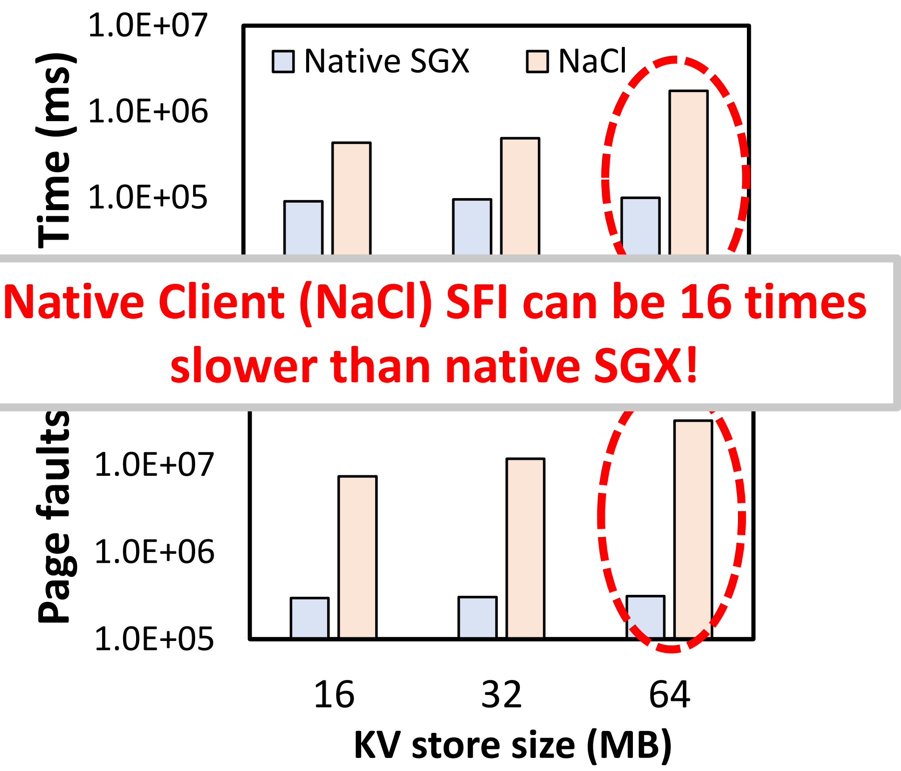
Must replicate common data in each process

High memory use severely reduces performance



Memory usage over 256 MB incurs expensive page faults

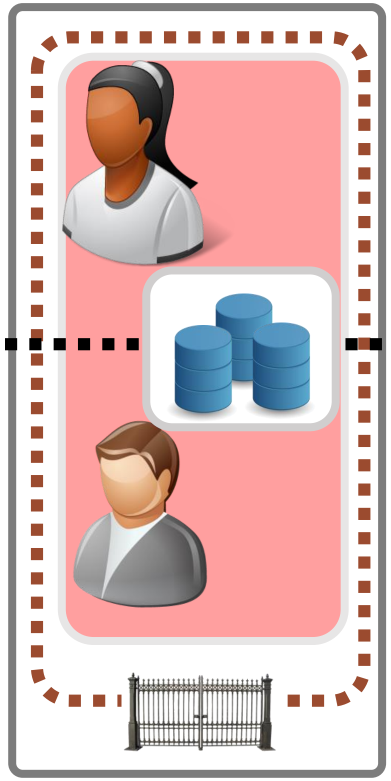
Key-value store with
8 clients



Native Client (NaCl) SFI can be 16 times slower than native SGX!

Chancel implements efficient multi-client SFI

Multiple clients are served within a process



Clients securely access shared memory

Chancel’s design

Offline stage

1. Automated program instrumentation

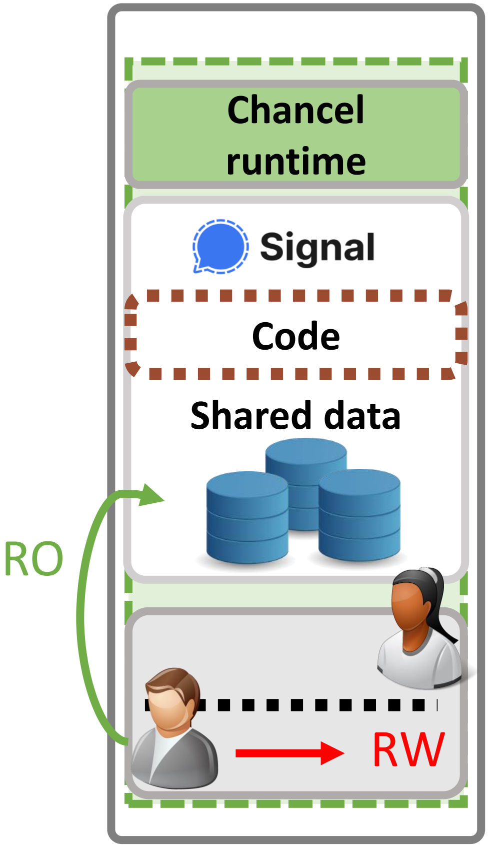
Chancel compiler

Online stages

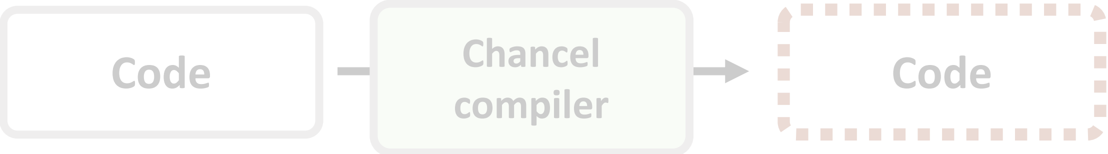
2. Enclave initialization and program loading

3. Secure client bootstrapping

4. Multi-client SFI enforcement



1. Automated program instrumentation

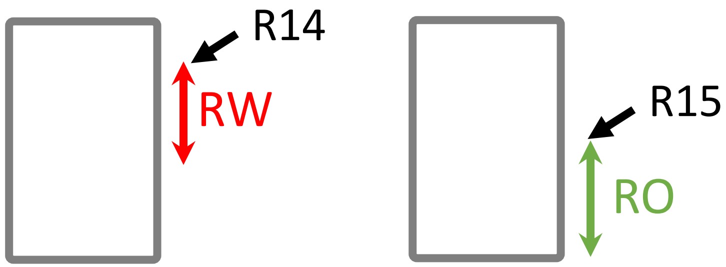


Native program code

Chancel binary

✗ R14 = …
R15 = …

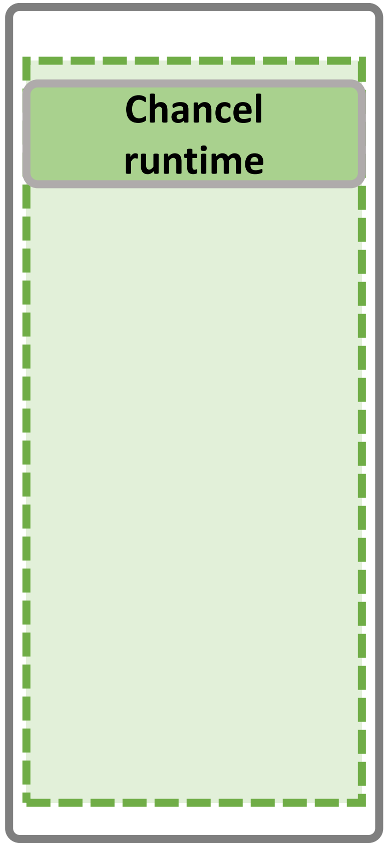
Compiler reserves registers R14 and R15



Compiler checks writes relative to R14 and reads relative to R14 or R15

2. Enclave initialization and program loading

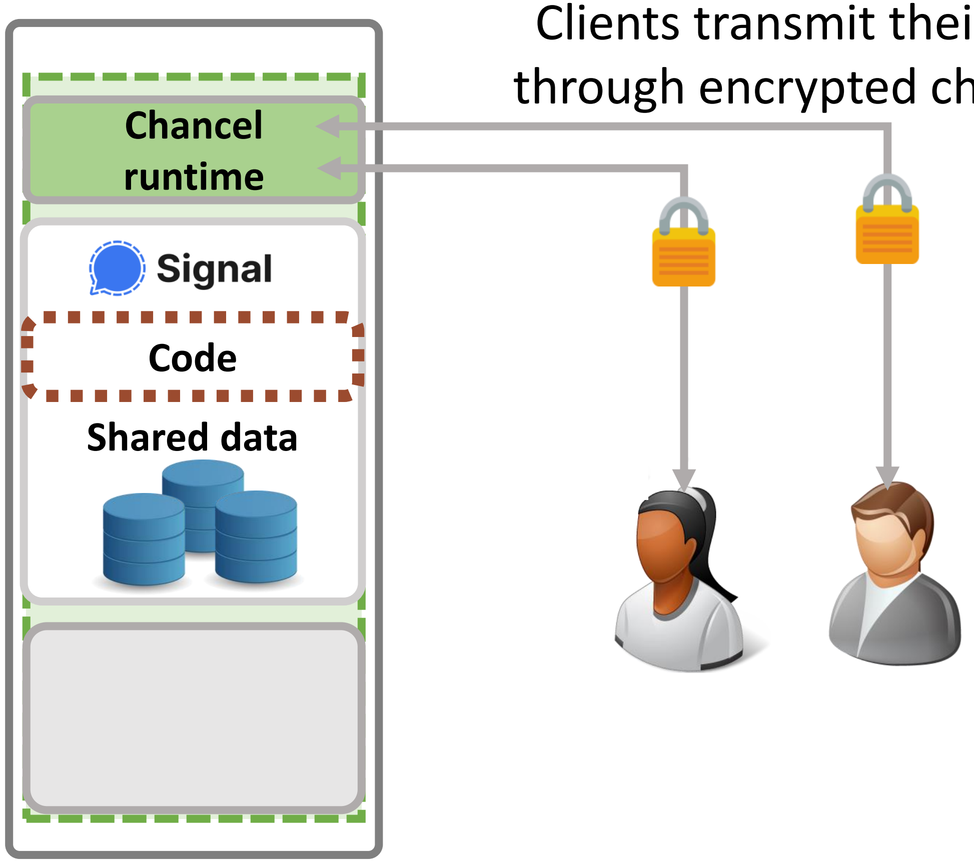
Validate instrumentation using a binary disassembler



Thanks to validation, Chancel even supports proprietary code!

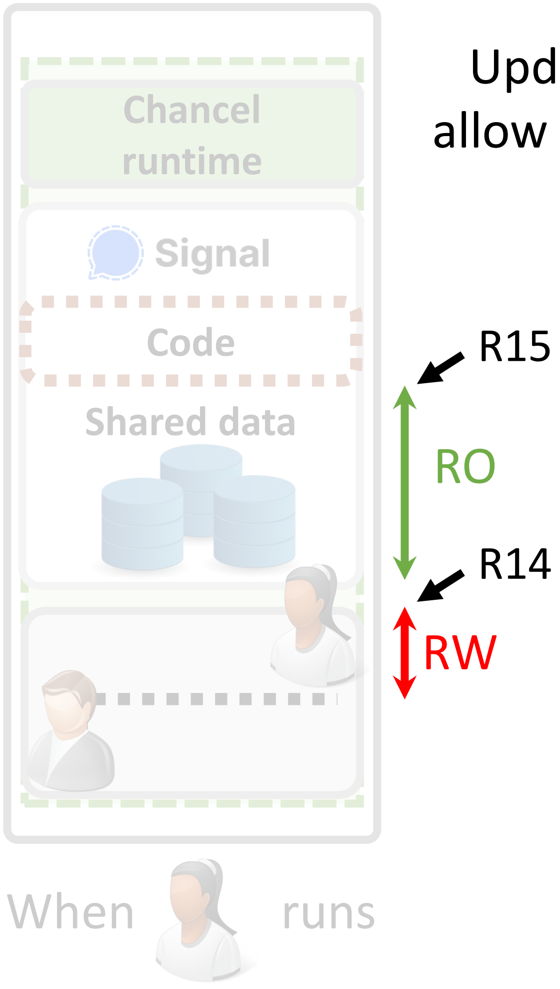
3. Secure client bootstrapping

Clients transmit their data through encrypted channels

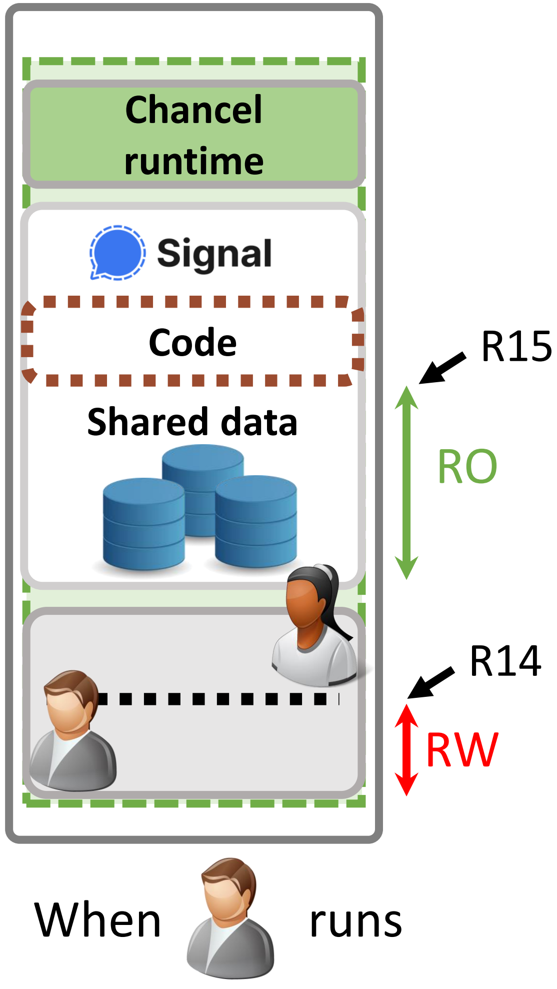


Store each client's data in a different enclave thread

4. Multi-client SFI enforcement

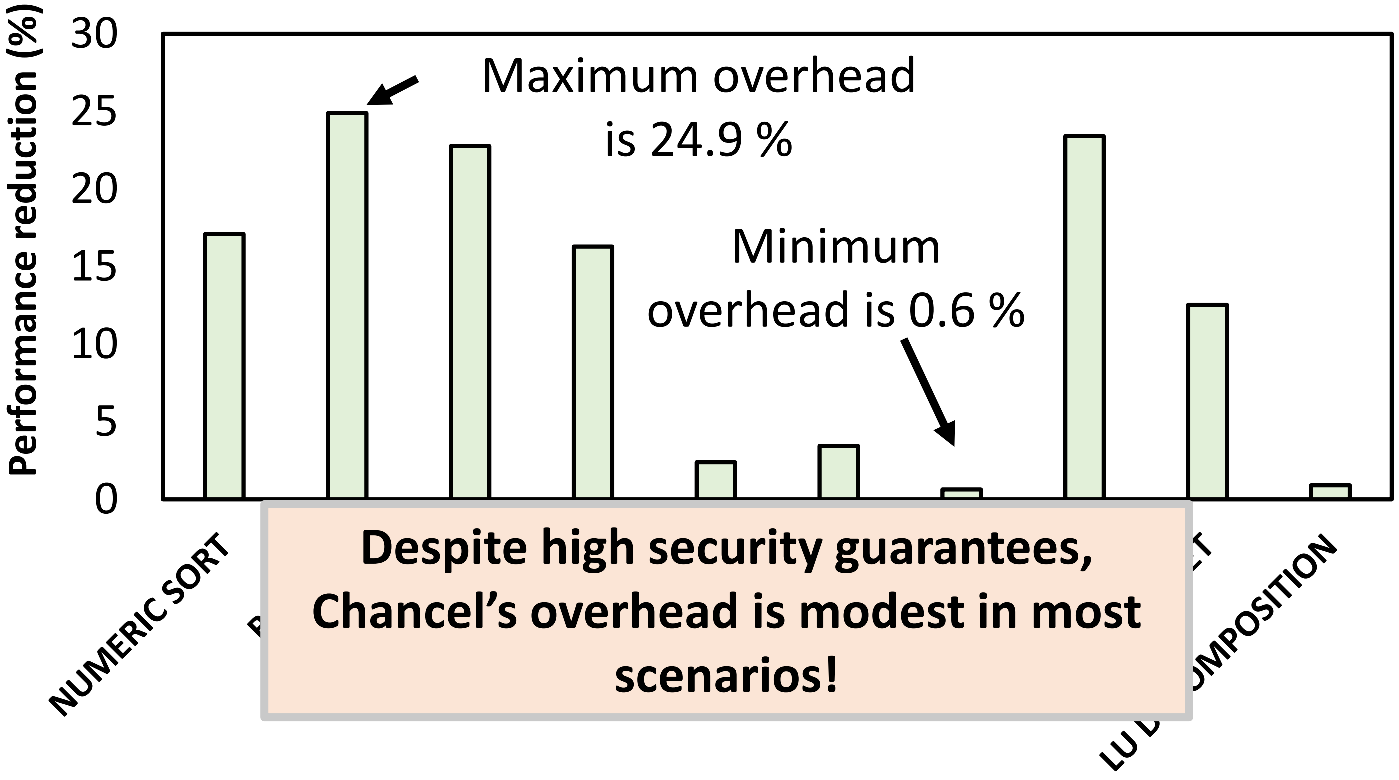


Update R14 and R15 to allow shared read but per-thread writes!



Overhead over native SGX

Ran all applications in Nbench, a popular SGX CPU and memory benchmark



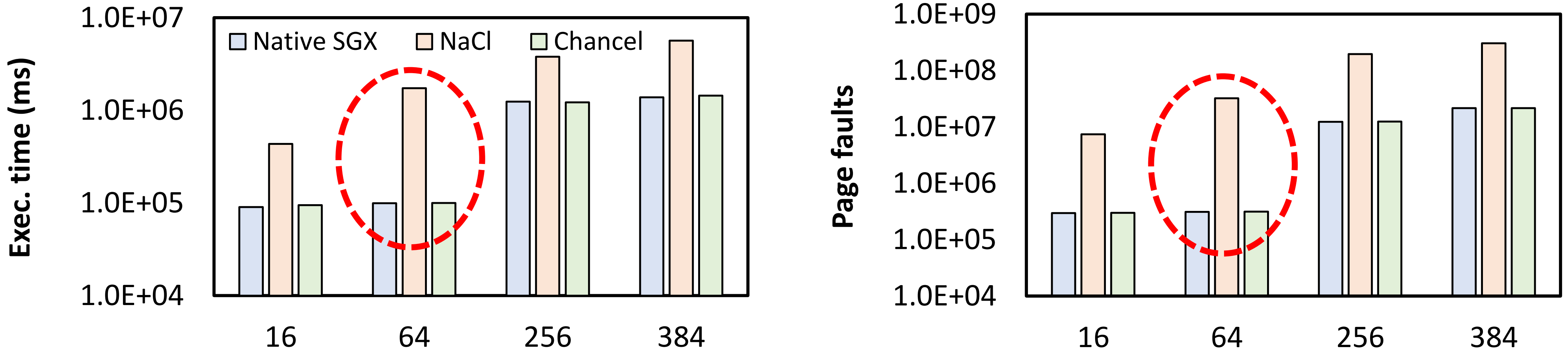
Maximum overhead is 24.9 %

Minimum overhead is 0.6 %

Despite high security guarantees, Chancel’s overhead is modest in most scenarios!

Benefit over Native Client

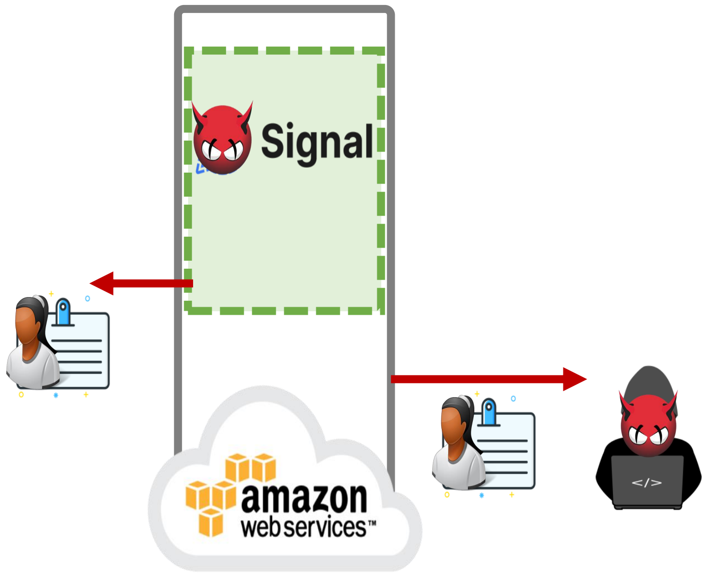
100,000 “GET” requests to ShieldStore key-value store from 8 clients



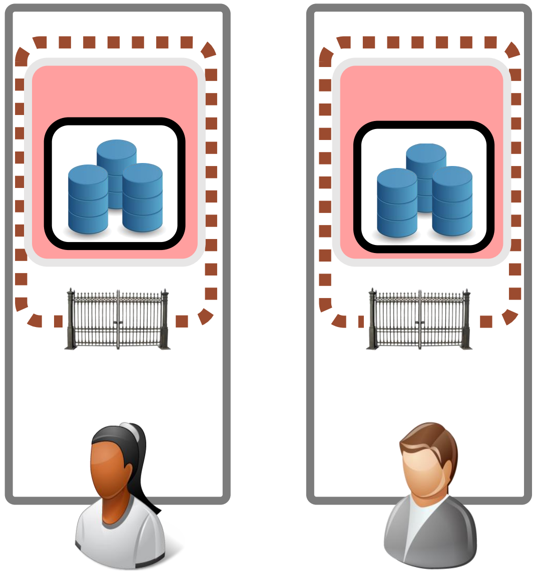
Across diverse applications, Chancel outperforms Native Client (NaCl) by up to 21 times!

Summary and conclusion

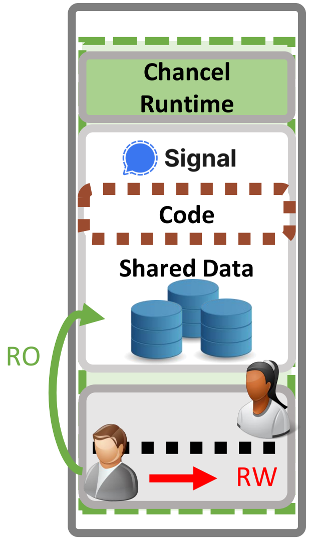
SGX does not secure remote data from untrusted code



Native Client (NaCl) SFI is slow in multi-client enclaves



Chancel's multi-client SFI is up to 21 times faster than NaCl



Thank you!