PRIDWEN
Universally Hardening SGX Programs via Load-Time Synthesis

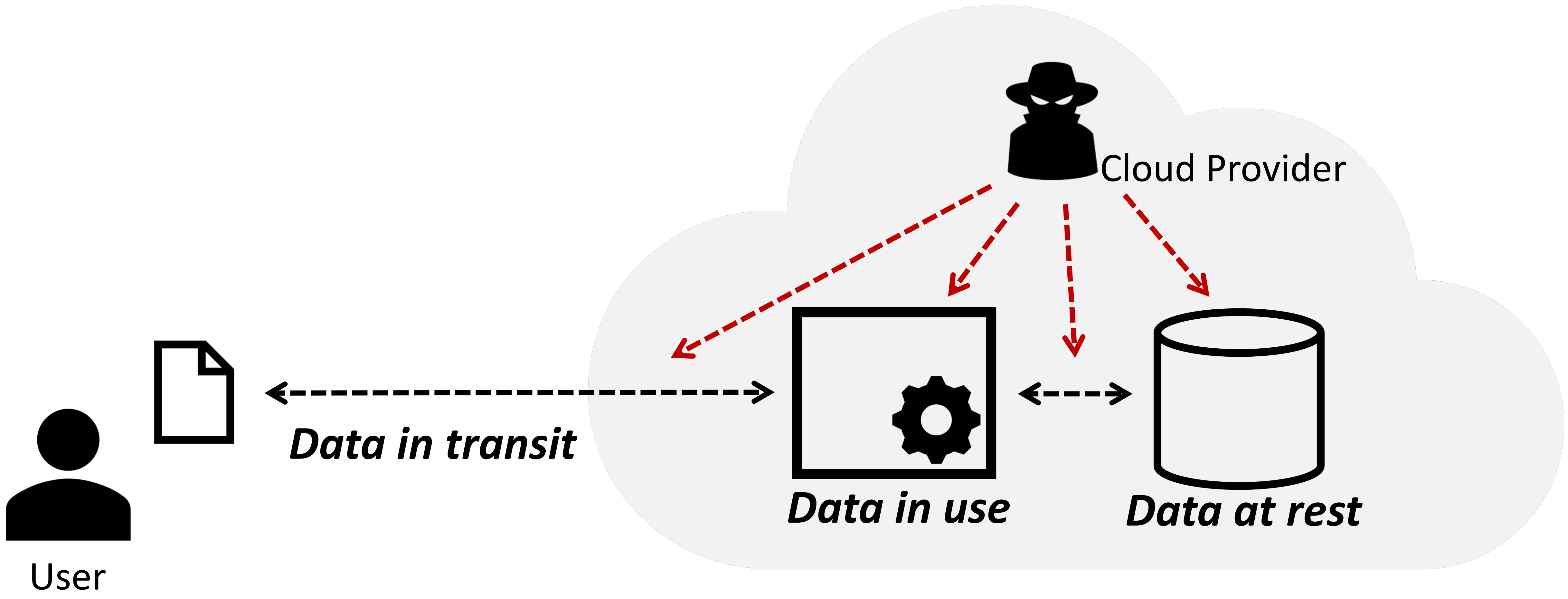
Fan Sang[*,1], Ming-Wei Shih[*,3], Sangho Lee[4], Xiaokuan Zhang[1], Michael Steiner[2], Mona Vij[2], Taesoo Kim[1]

[1]Georgia Institute of Technology, [2]Intel Labs, [3]Microsoft, [4]Microsoft Research

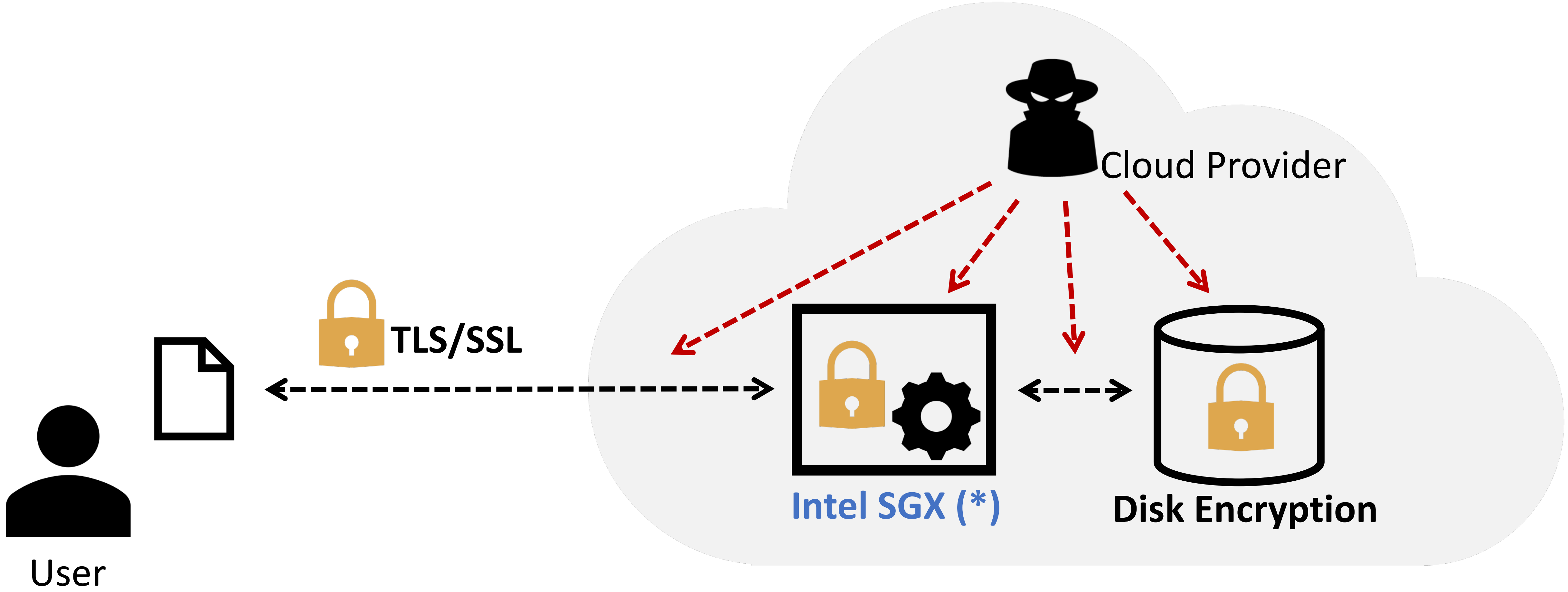
[*]Authors contributed equally to this work.

Prevalence of Cloud Computing Today

Concerns with Cloud: Data Security



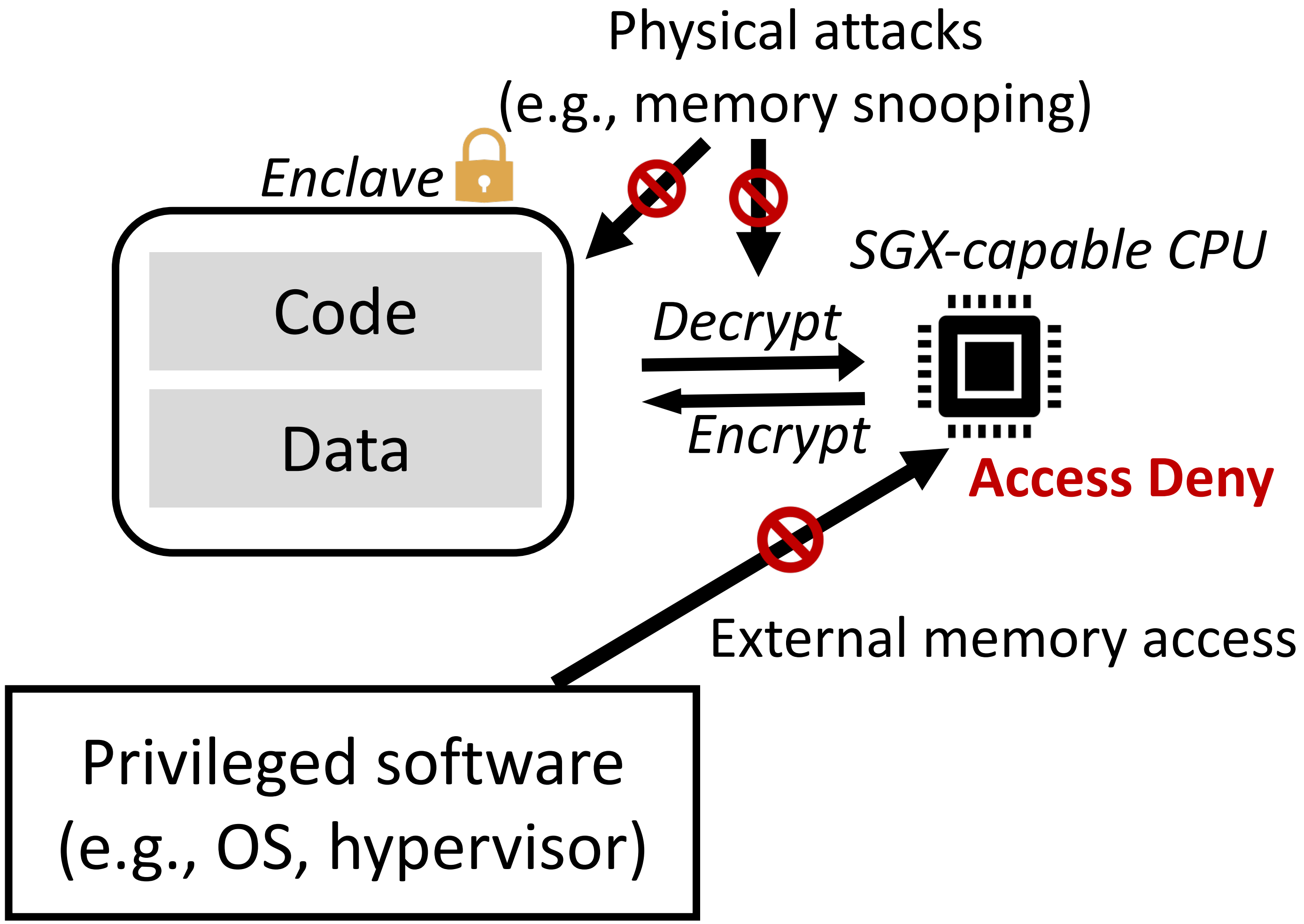
Existing Security Solutions



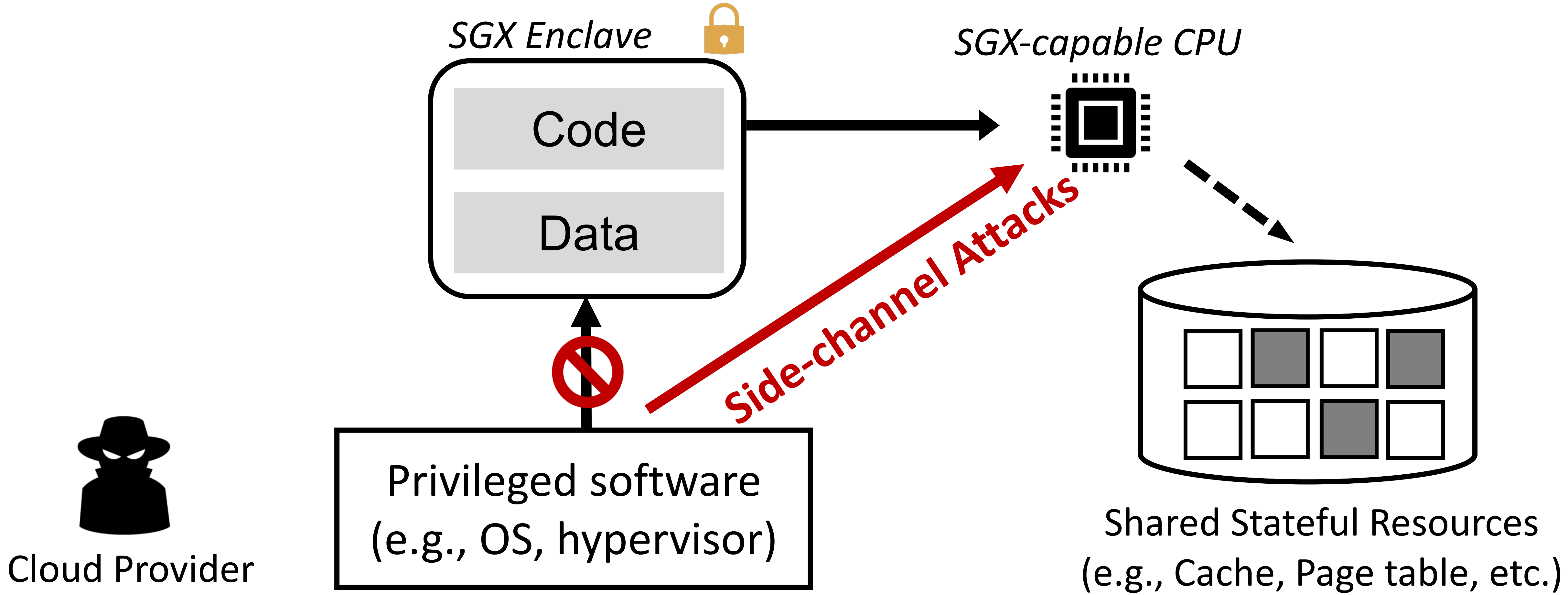
*Intel Software Guard Extension (SGX)

Intel SGX 101

- ***Enclave***: Isolated memory region
  - Strict memory access control
  - Memory encryption
- **Remote attestation**
  - Allows for attesting code/data inside a remote enclave



Achilles' Heel of SGX: Side-Channel Attacks



Cloud providers as attackers (with root privilege)

- ***Side-channel inference with low-noise, high-resolution***

Side-Channel Attacks Against SGX

- Shared resources as side channels
  - **Page table** [*SP’15, Security’17*]
  - **Cache** [*WOOT’17, ATC’17, CHES’17*]
  - **Branch predictor** [*Security’17, ASPLOS’18*]
  - **TLB** [*CCS’17, Security’18*]
  - **CPU pipelines** [Security’18, EuroSP’19, SP’20, SP’21]
- Allow the attacker to infer fine-grained information inside the enclave

→ **Break the security guarantees of SGX**

Question: How to address the side-channel attacks against SGX?

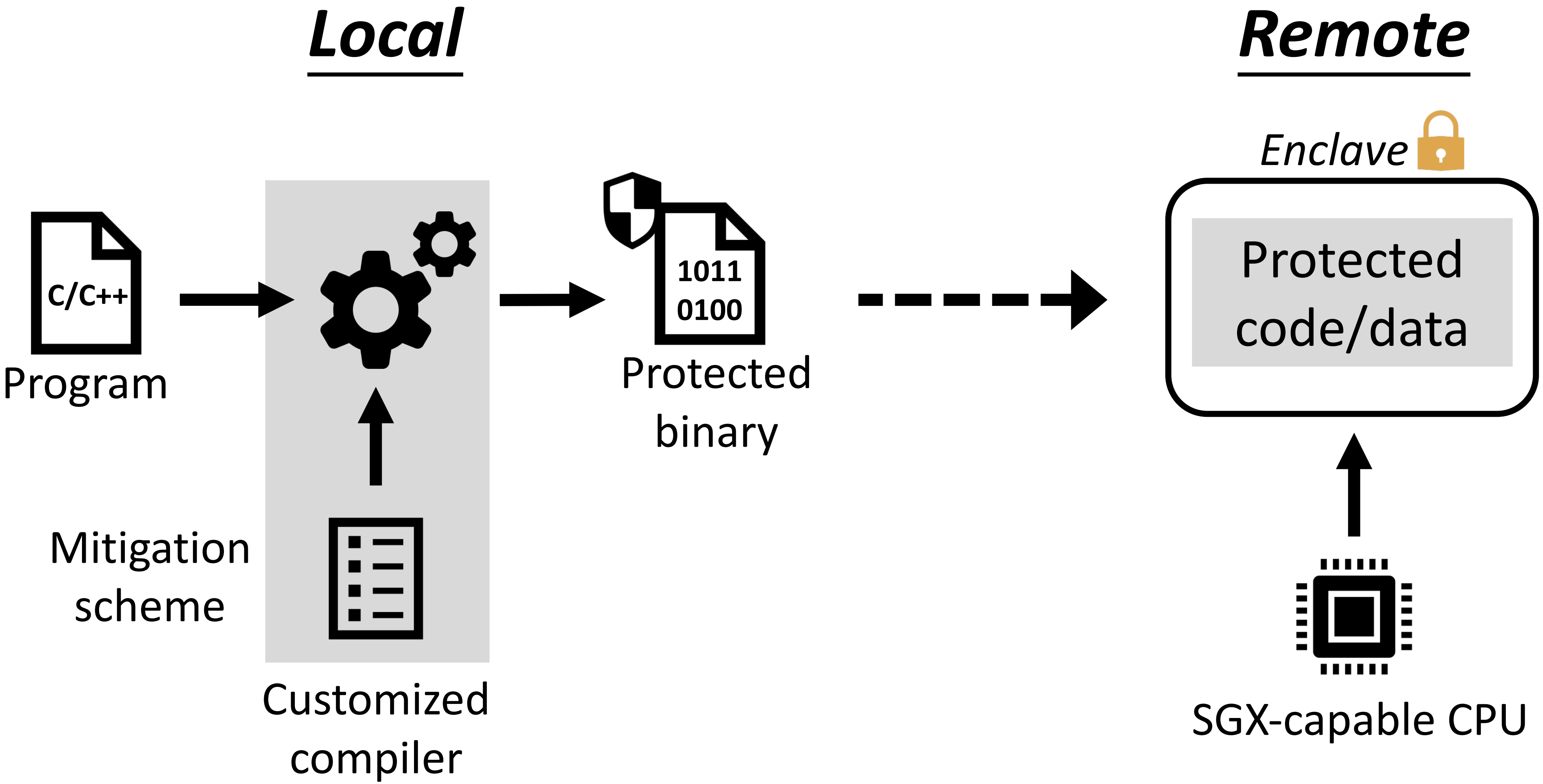
Side Channel Mitigation Schemes

Scheme	Mitigation Target
SGX-Shield [*NDSS'17*]	Fine-grained ASLR
Varys [*ATC'18*]	High-frequent interrupt-based attacks
T-SGX [*NDSS'17*]	Page-fault attacks
Cloak [*Security'17*]	Cache attacks
HyperRace [*DSC'19*]	Hyperthread-based attacks
Retpoline & Qspectre [*2018*]	Spectre attacks

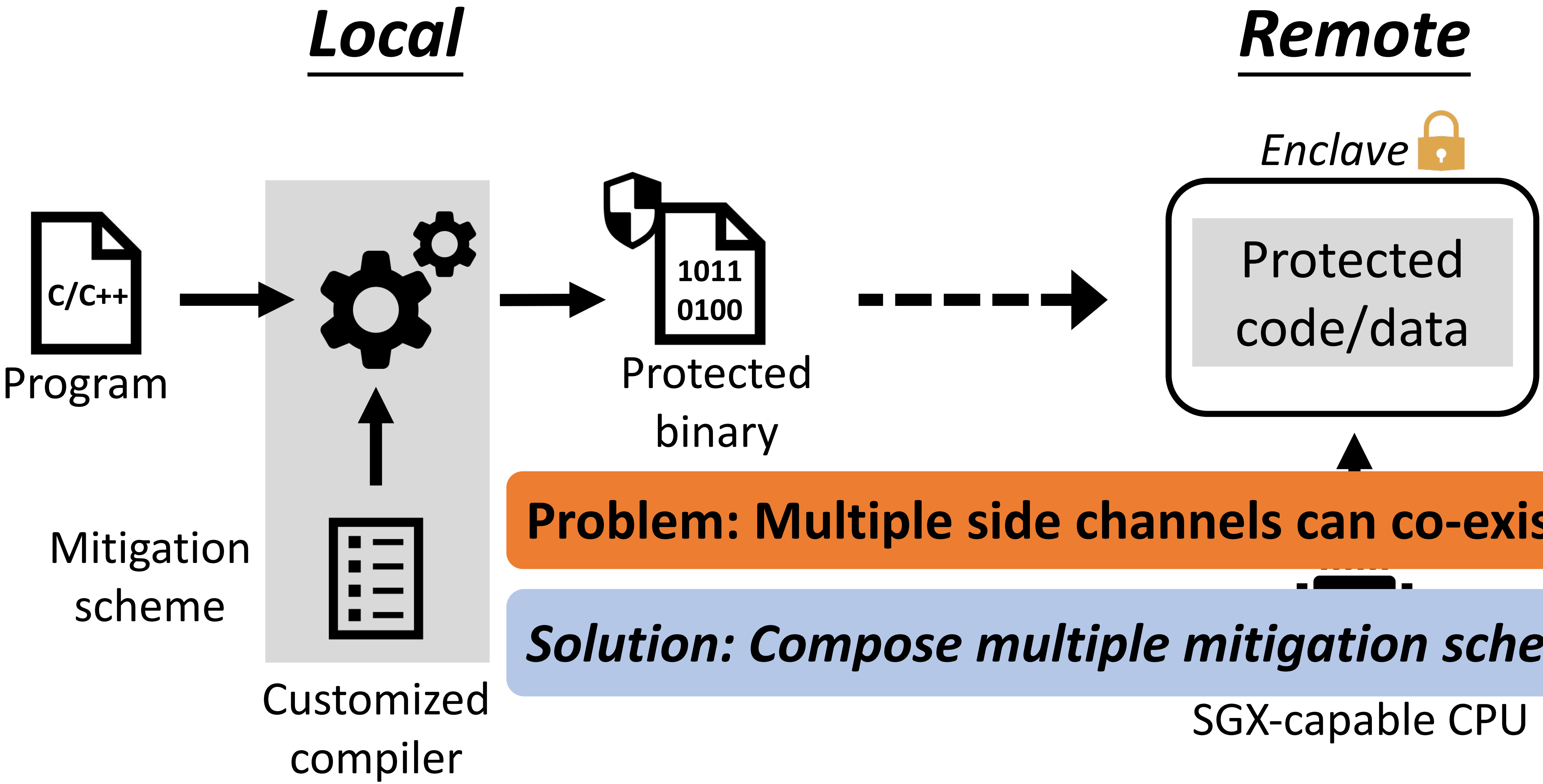
Similar design choices

- **Require no hardware modification**
- **Minimum manual efforts (*instrumentation-based*)**

Deployment of a Mitigation Scheme



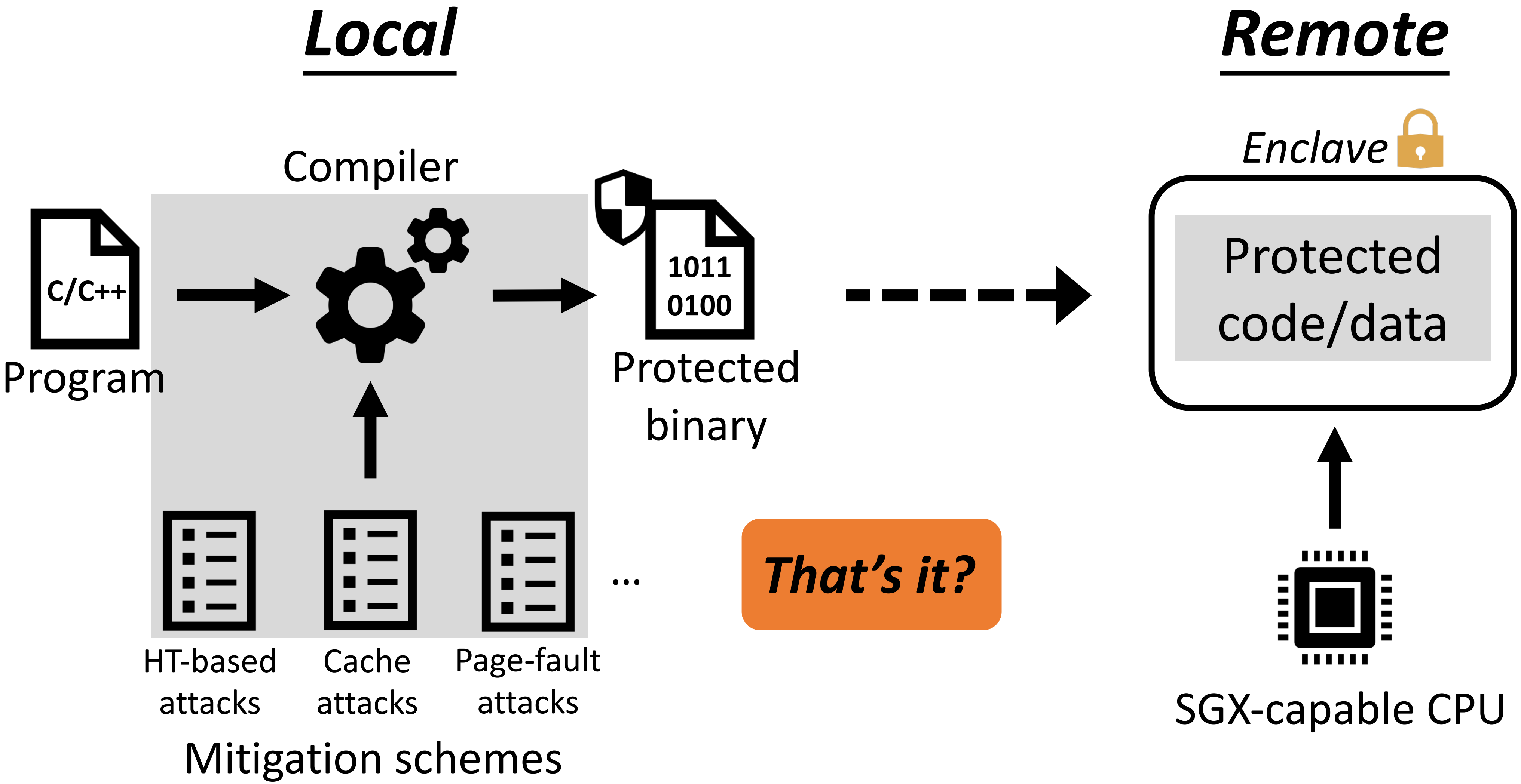
Each Scheme Targets Limited Types of Attacks



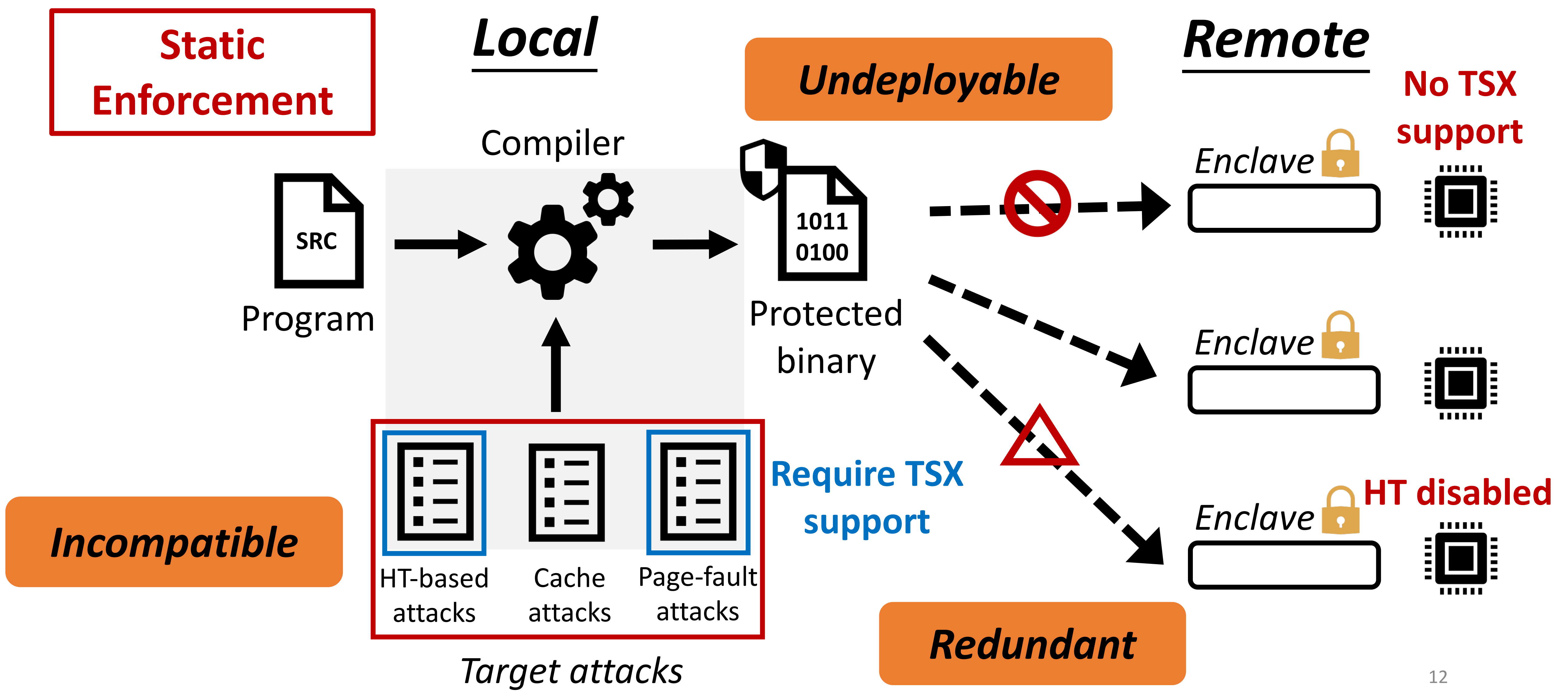
Problem: Multiple side channels can co-exist

Solution: Compose multiple mitigation schemes

Composing Multiple Mitigation Schemes



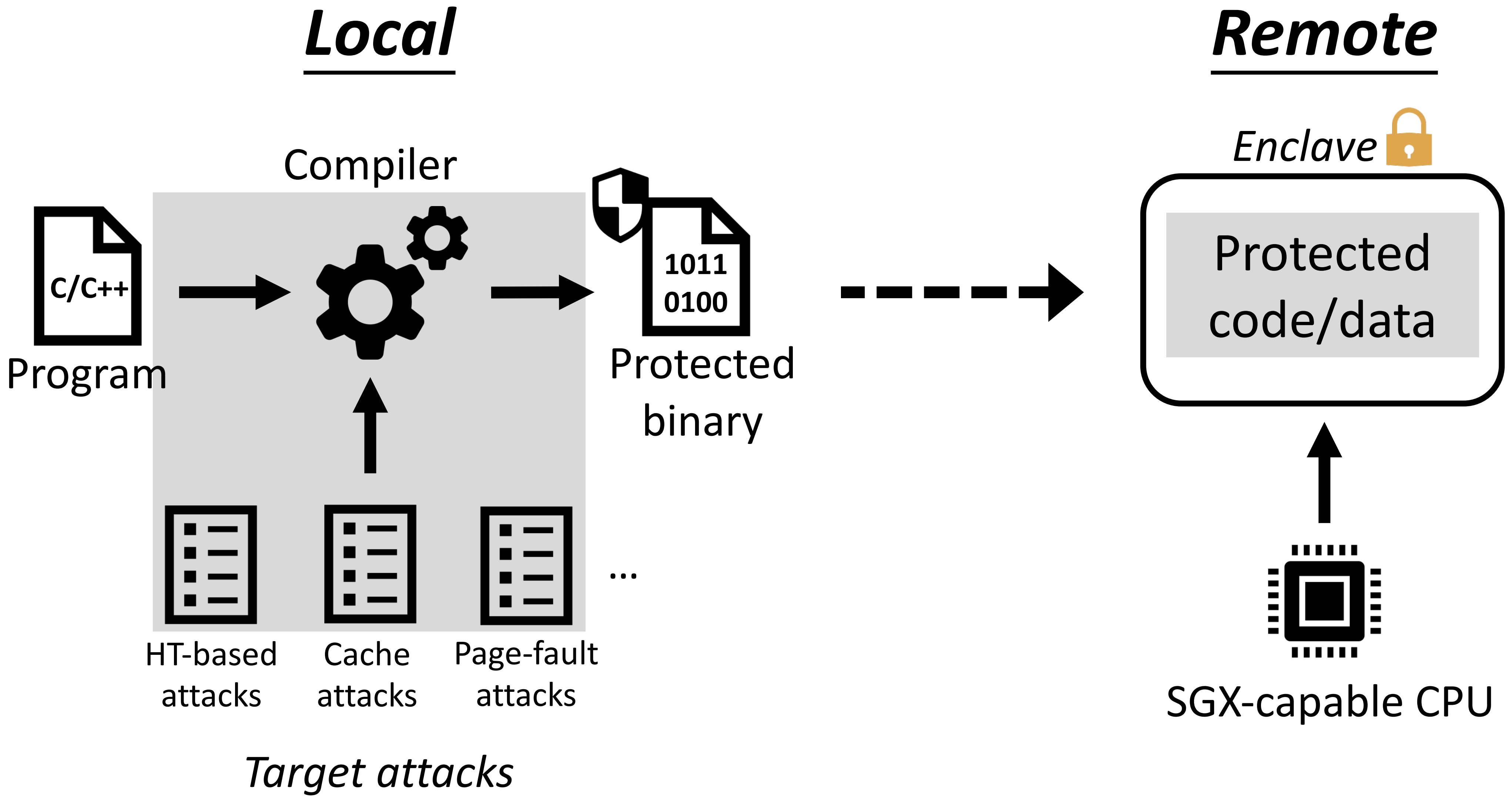
Problems with Naïve Scheme Composition



When Can We Make the Best Decisions?

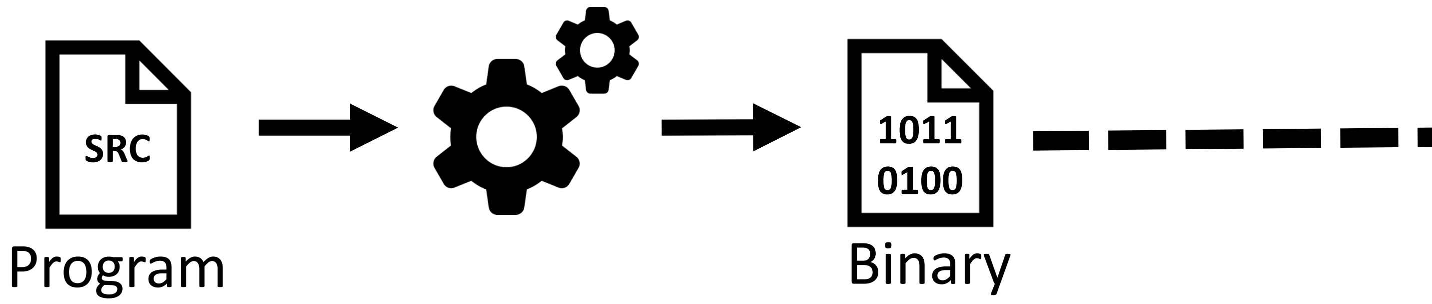
As Close to the Final Execution as Possible!

Local Scheme Enforcement

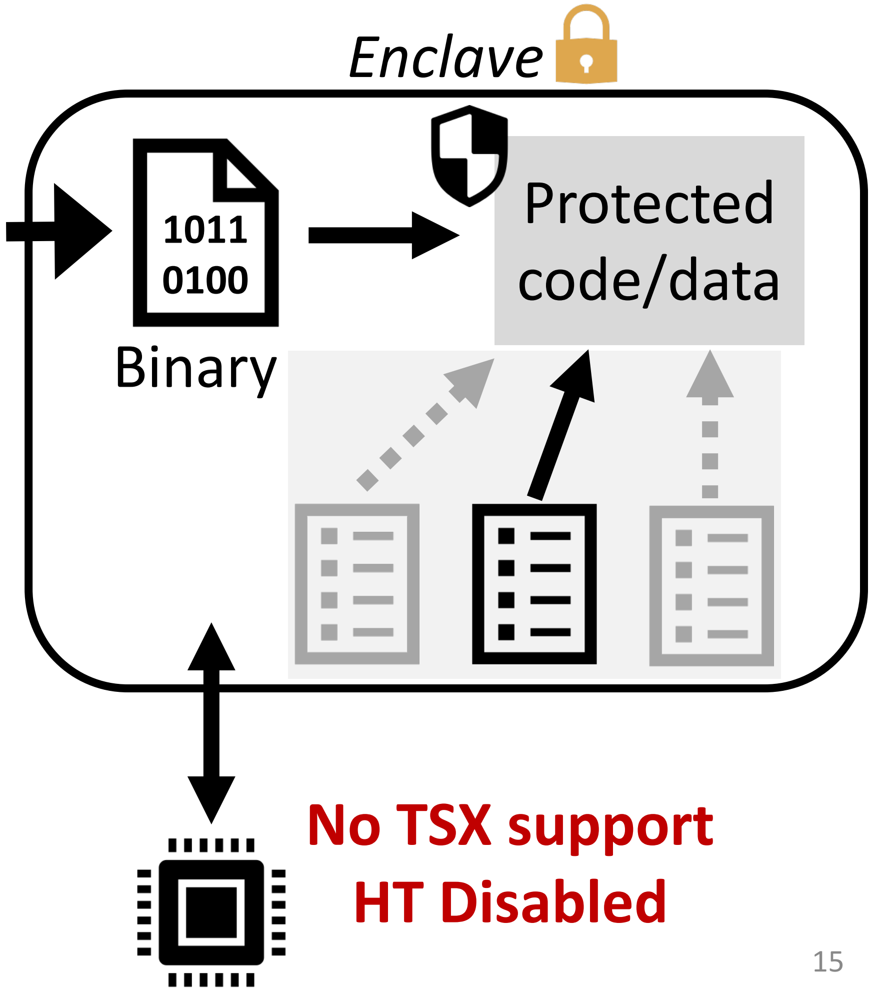


Post-Deployment Scheme Enforcement

Local



Remote

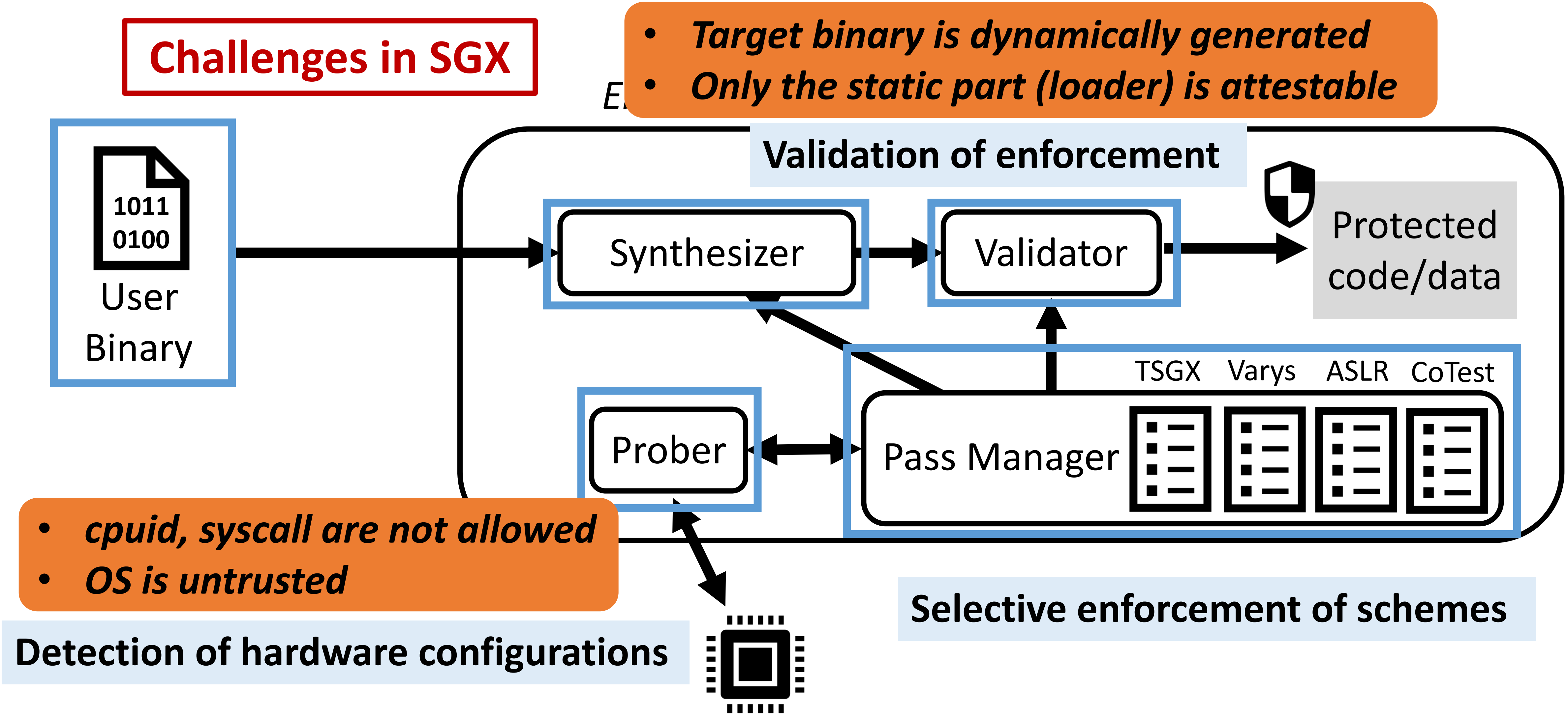


Requirements

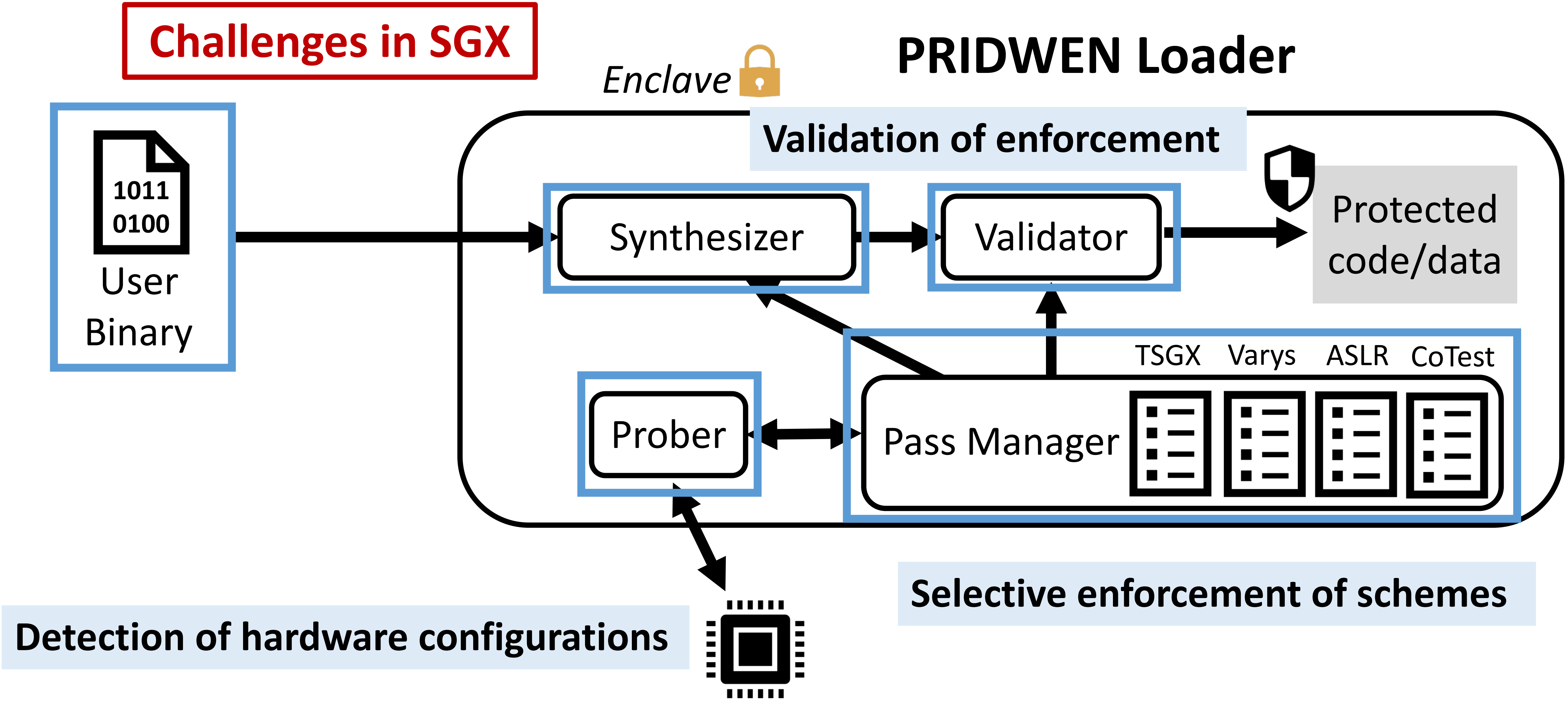
- **Detection of hardware configurations**
- **Selective enforcement of schemes**
- **Validation of enforcement**

PRIDWEN

PRIDWEN Overview



PRIDWEN Overview



Selective Enforcement of Schemes

- Approach: *Load-time synthesis*
  - Take the intermediate representation (IR) of a program as input
  - Support compilation and instrumentation of the IR
  - Provide APIs for implementing schemes as instrumentation passes

Use IR as Input

- Advantages over native binary
  - Friendly for code analysis and instrumentations
  - Platform independent
- IR selection: WebAssembly (WASM)
  - Lightweight (small instruction set), small TCB
  - Supports multiple high-level languages (e.g., C/C++, Rust)
  - Straightforward compilation

	Line of Code	Binary Size (MiB)
PRIDWEN backend	8,166	1.26
LLVM x86 backend	80,449	1,026.00

C to WebAssembly

C Language

```
int foo(int x) {
  if (x != 0) {
    return x * x;
  }
  return 0;
}
```

Compilation →

1011
0100

WASM binary

Decoding →

WASM IR

```
(func (;0;) (param i32)
  result i32)
  local.get 0
  local.get 0
  i32.mul
  local.set 1
  local.get 0
  if (result i32)
    local.get 1
  else
    i32.const 0
end)
```

Supported by opensource compiler
Require no source-code modifications

Supported by PRIDWEN

Flexible Instrumentation

IR-level

```
(func (;0;) (param i32)
  result i32)
  local.get 0
  local.get 0
  i32.mul
  local.set 1
  local.get 0
  if (result i32)
    local.get 1
  else
    i32.const 0
end)
```

Hook before
`+local.get 0`

Hook after
`+i32.add`

Native-level

```
     push rbp
     mov rbp, rsp
     sub rsp, 0x10
     mov eax, edi
     imul eax, edi
     test edi, edi
     je  $1 ; else
     jmp $2
$1:  xor eax, eax
$2:  mov rsp, rbp
     pop rbp
     ret
```

Hook before
`+mov r15, $1`

Hook after
`+lfence`

PRIDWEN Instrumentation APIs

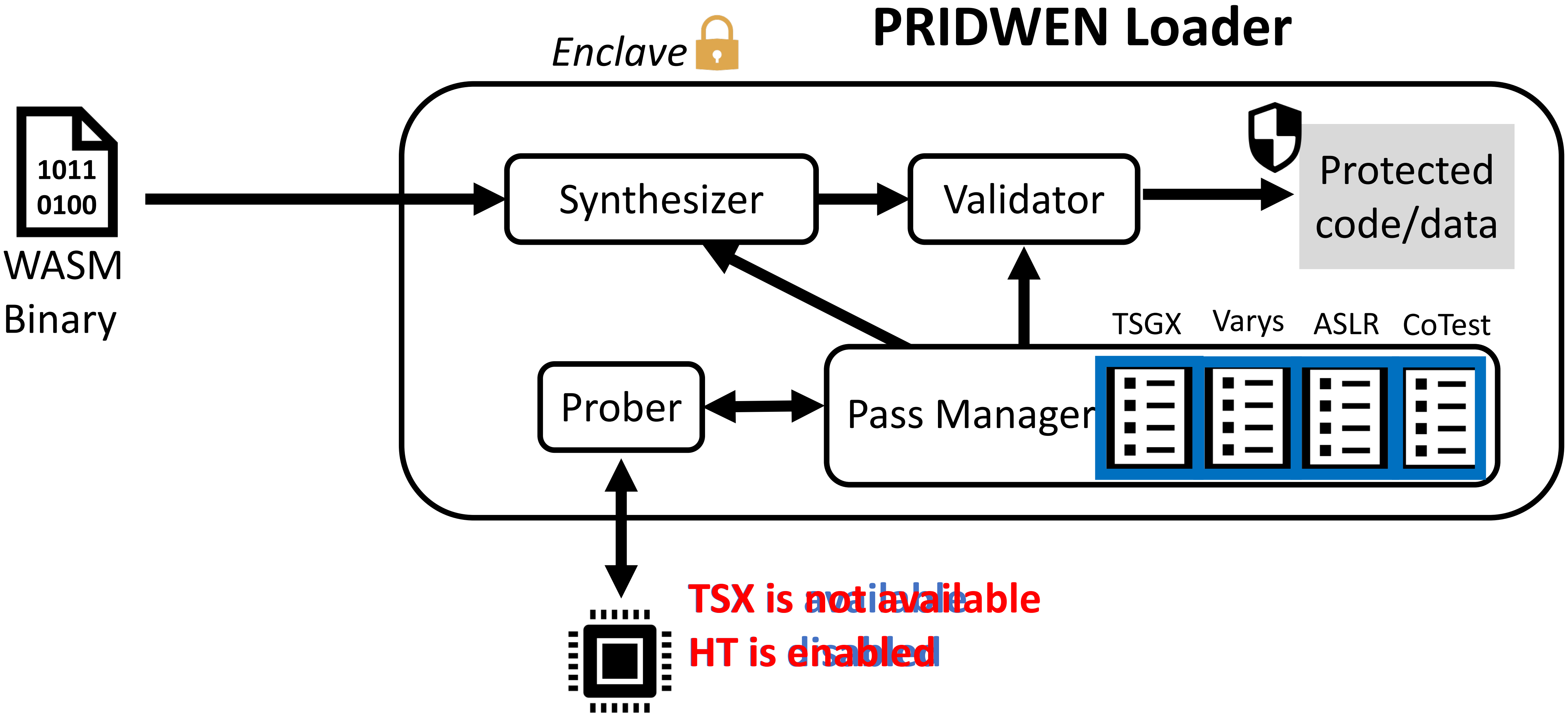
- IR-level

```
onFunctionStart(CompilerContext *ctx)
onFunctionEnd(CompilerContext *ctx)
onControlStart(CompilerContext *ctx)
onControlEnd(CompilerContext *ctx)
onInstrStart(CompilerContext *ctx)
onInstrEnd(CompilerContext *ctx)
```

- Native-level

```
onMachineInstrStart(CompilerContext *ctx, MachineInstr *mi)
onMachineInstrEnd(CompilerContext *ctx, MachineInstr *mi)
```

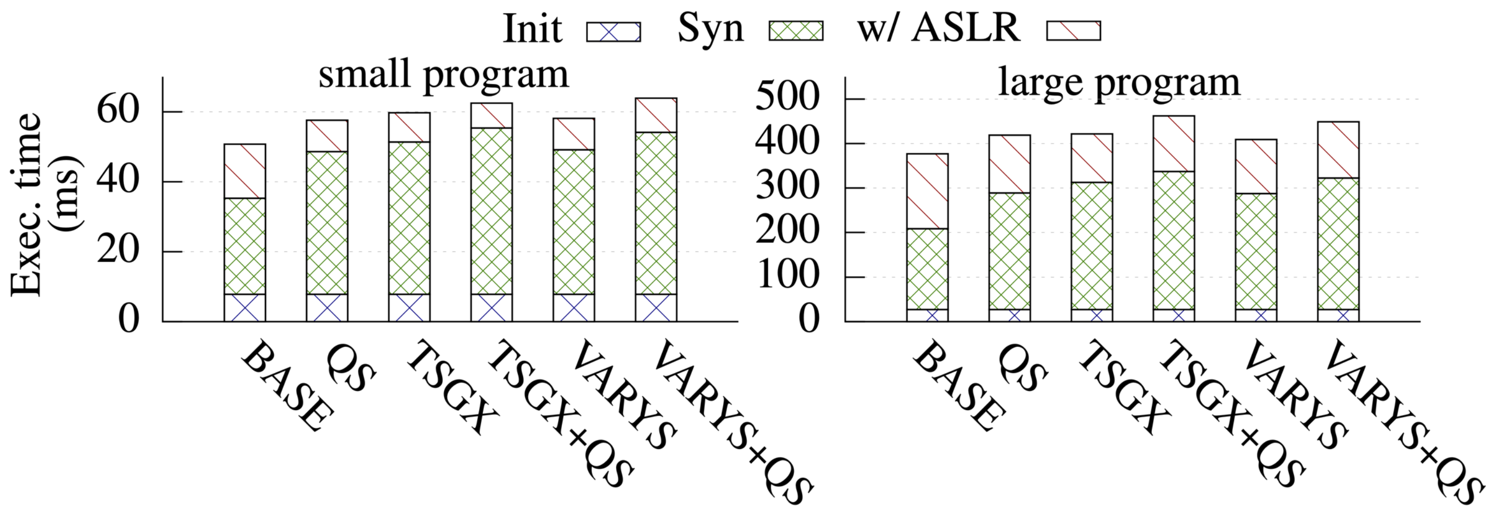
PRIDWEN In Action



Evaluation: Overhead of PRIDWEN Loader

- The performance of synthesizing small and large programs
  - Small program (~50 kB): **50 - 60 ms**
  - Large program (~500 kB): **< 500 ms**

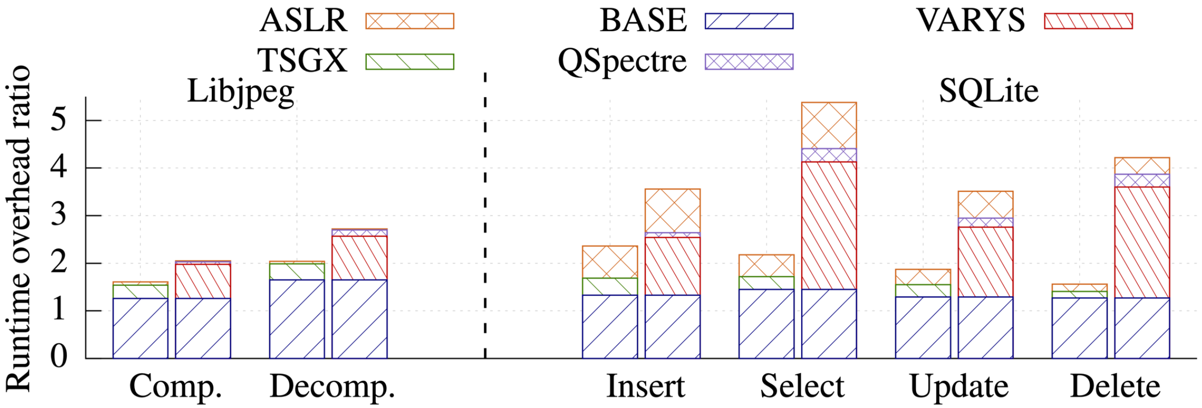
Paid only once



Evaluation: Baseline Runtime Performance

- Relative runtime performance of PRIDWEN-synthesized applications compared to native versions
  - Lighttpd: **1.5x**
  - SQLite: **1.3x**
  - libjpeg: **1.4x**
  - Recent study shows in-browser WASM JITs on SPEC: **1.45x – 1.55x**

Evaluation: Overhead of Mitigation Schemes



- The performance of libjpeg and SQLite
  - HW-assisted: **1.9x**
  - SW-only: **3.4x**

Comparable to the original implementations

Conclusion

- SGX side-channel attacks can co-exist
- Existing model for deploying mitigation schemes is limited
- We propose **PRIDWEN** to achieve scheme composition
  - Detect hardware configurations
  - Adaptively enforce mitigation schemes with an in-enclave loader
  - Extensible framework to support more schemes

https://github.com/sslab-gatech/Pridwen

Q&A

Thank You!

Fan Sang[†,1], Ming-Wei Shih[3], Sangho Lee[4], Xiaokuan Zhang[1],
Michael Steiner[2], Mona Vij[2], Taesoo Kim[1]

[1]Georgia Institute of Technology, [2]Intel Labs, [3]Microsoft, [4]Microsoft Research

[†]fsang@gatech.edu

SSLab @GeorgiaTech **Systems Software & Security Lab**

Georgia Tech College of Computing
School of Cybersecurity and Privacy