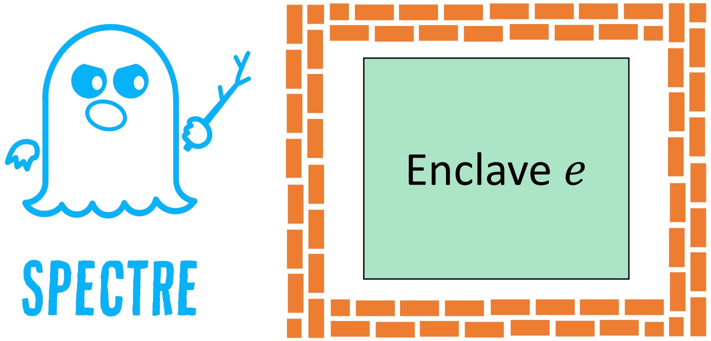
A Formal Approach to Secure Computation

(Presented by Kevin Cheang)

ADEPT End-of-Project Party
Dec 9th 2021

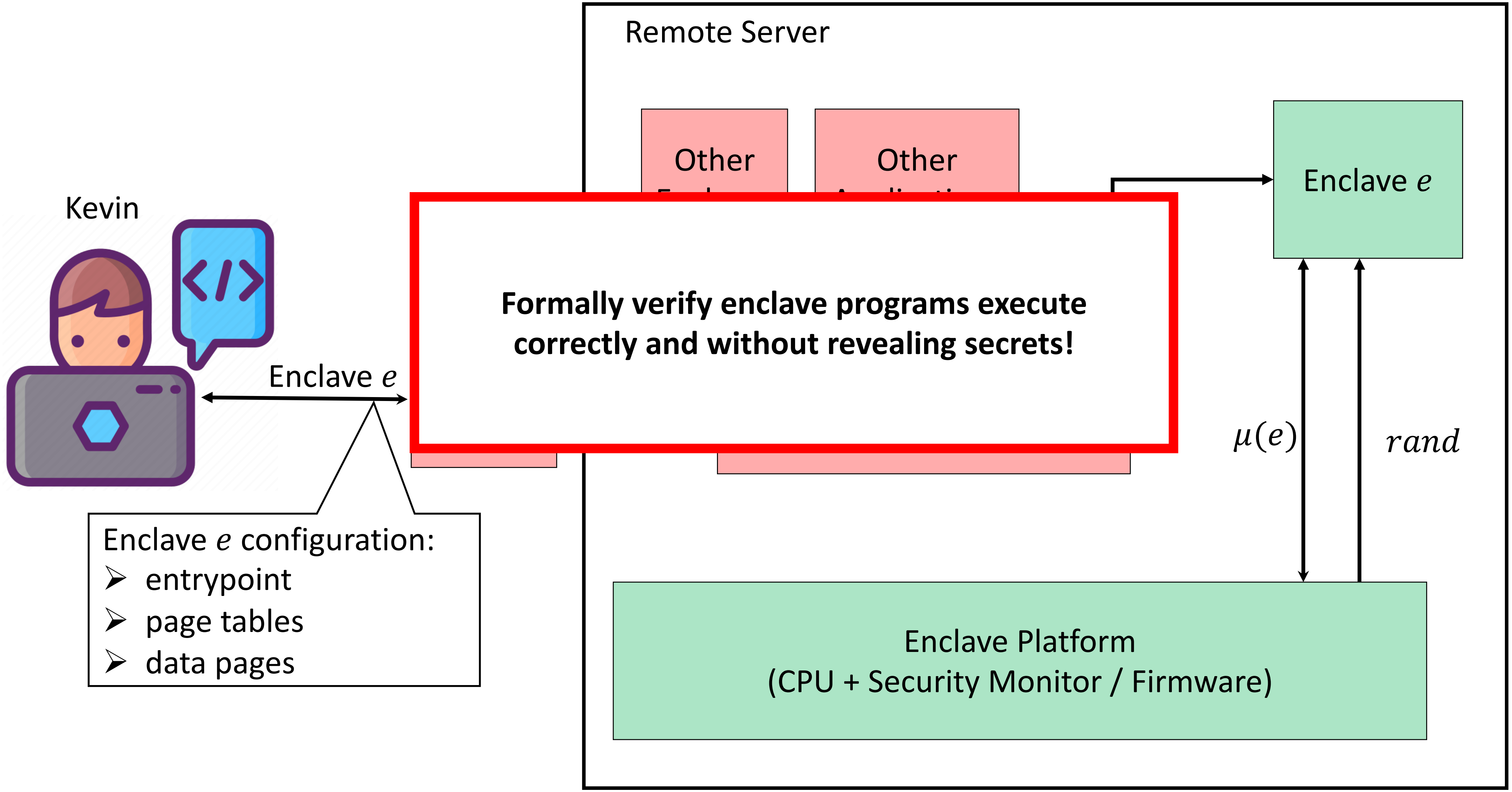
Demand for Secure and Remote Computation

- Strong memory isolation through enclaves
  - An enclave is a program that provides memory isolation typically using hardware primitives and a software interface



- Why do we need them?
  - Remote computation on the cloud is very common (e.g. AWS)
  - Want our data to be kept secret
  - Recent attacks: Spectre, Spectre-NG, Meltdown, Foreshadow, Fallout, …

Trusted Execution Environments



A Formal Foundation for Secure Remote Execution of Enclaves

Pramod Subramanyan, Rohit Sinha, Illia Lebedev,
Srinivas Devadas, Sanjit A. Seshia

CCS 2017

A Formal Foundation for the Secure Execution of Enclaves

- **Goal**: Introduce a novel formal verification methodology based on a trusted abstract platform to model enclaves and to prove secure remote execution on enclave platforms

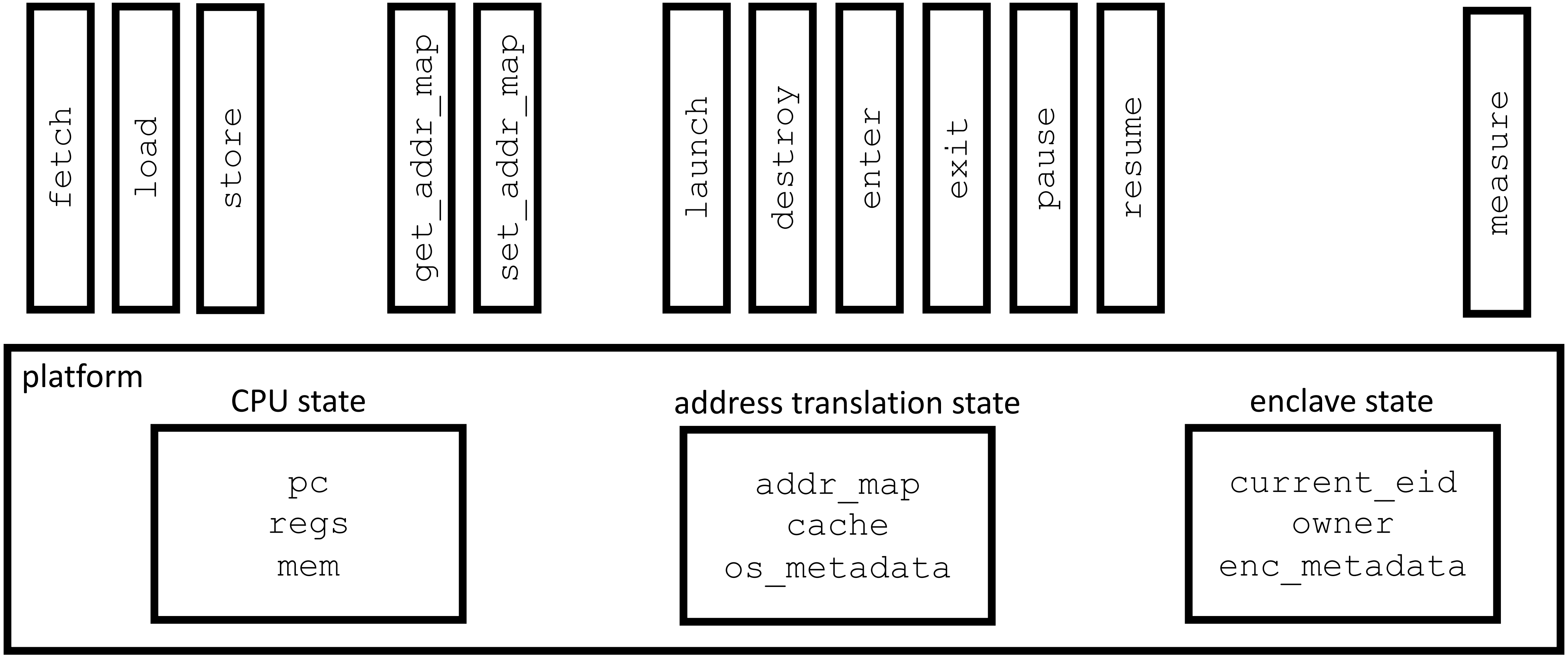
- **Contributions**:
  - Formalization of enclave execution against a privileged adversary
  - Formalization of secure remote execution (SRE)
  - Decomposed SRE into secure measurement, integrity and confidentiality
  - Modeled the trusted abstract platform (TAP) and proved SRE on TAP
  - Refinement based methodology; showed SGX and Sanctum refines TAP

Secure Remote Execution

SRE Definition: *A remote platform performs secure execution of enclaves if the execution follows the expected semantics and does not reveal anymore than what is allowed by the adversary's observations.*

Theorem 3.2*: An enclave platform that satisfies secure measurement, integrity and confidentiality property for any enclave program also satisfies secure remote execution.*

The Trusted Abstract Platform



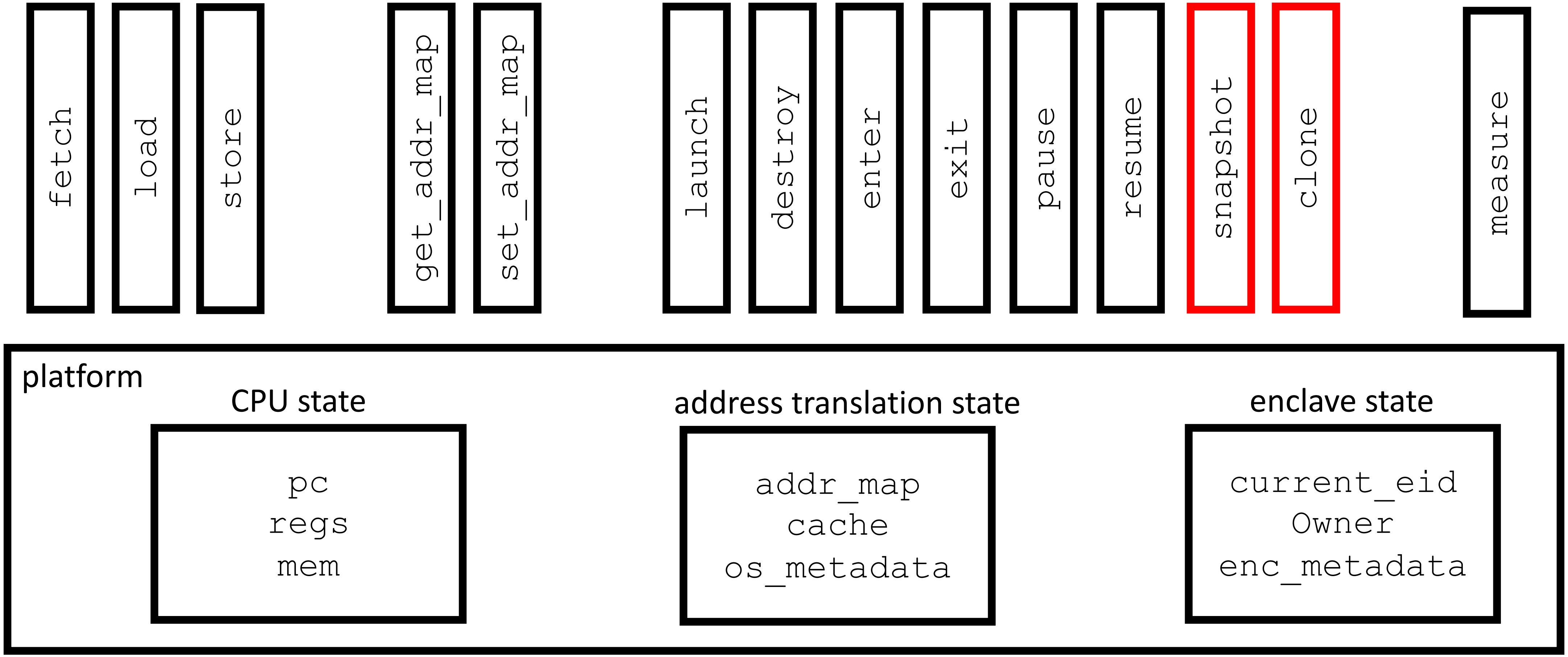
A Formal Approach to Secure and Efficient Enclave Cloning

Dayeol Lee, Kevin Cheang, Alexander Thomas, Catherine Lu, Pranav Gaddamadugu, Anjo Vahldiek-Oberwagner, Mona Vij, Dawn Song, Krste Asanovic, Sanjit A. Seshia

[in submission at S&P 2022]

The Trusted Abstract Platform with Cloning

(recently submitted to S&P 2022!)



A Formal Approach to Secure Speculation

Kevin Cheang, Cameron Rasmussen, Sanjit A. Seshia

Pramod Subramanyan

CSF 2019

A Formal Approach to Secure Speculation

- **Goal**:
  - Verify that programs are not vulnerable to transient execution attacks running on a given microarchitectural design
- Mitigations
  - Software mitigations
    - compiler extensions (e.g. /QSpectre), retpolines, page table isolation
  - Hardware migitations
    - constant time memory loads, cache partitioning, access time randomization, obfuscation of timers
- **Contributions**
  - Formulating a general property to capture transient execution attacks
  - Introduced an assembly intermediate representation for speculative platforms
  - Automated verification of absence of transient execution attacks

Problem Statement

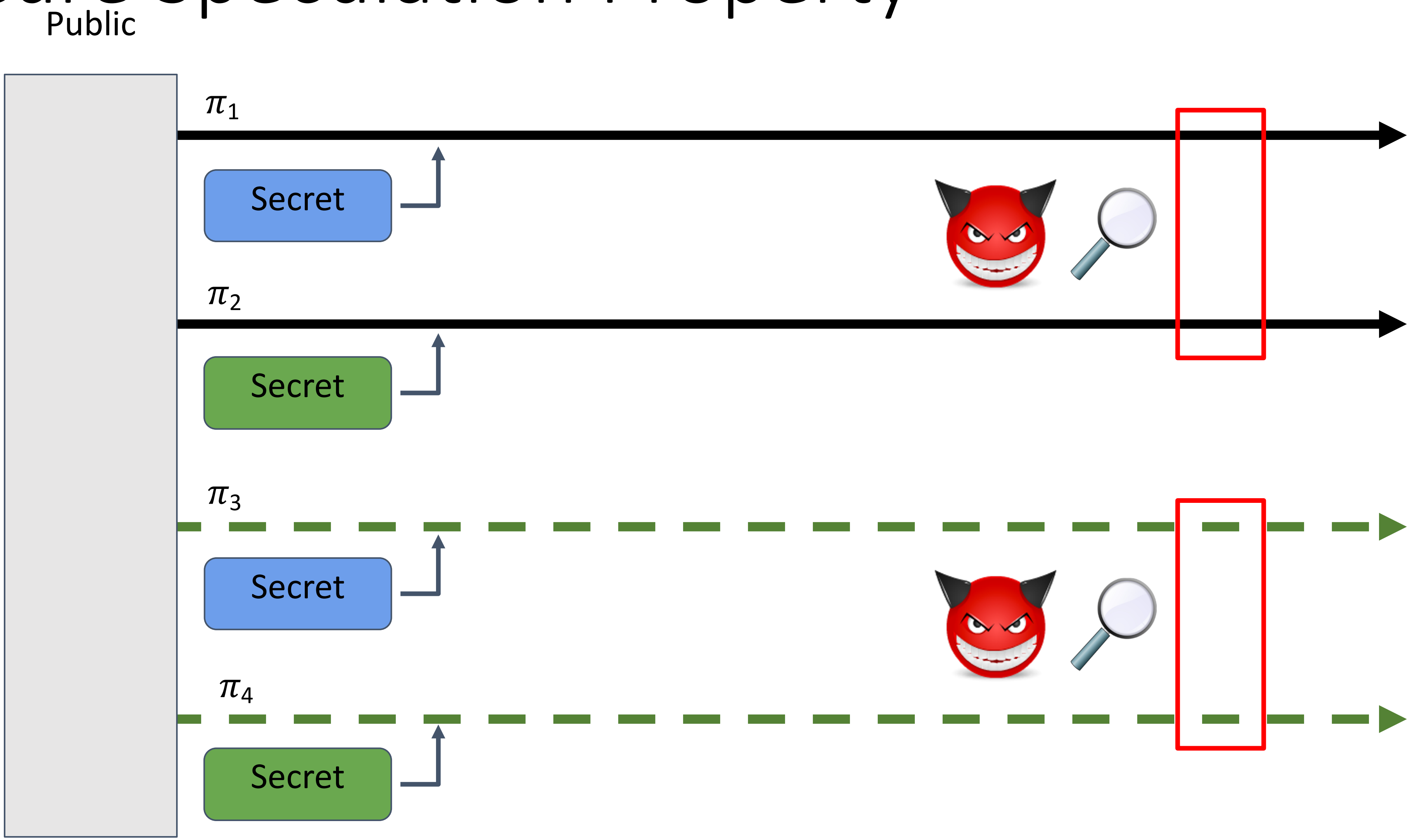
- Given a platform model, an adversary model, and a program, determine if the program is vulnerable to a transient execution attack

```
void victim_function_v01(size_t x) {
     if (x < array1_size) {
          temp &= array2[array1[x] * 512];
     }
}
```

Spectre Variant 1

```
void victim_function_v01(size_t x) {
     if (x < array1_size) {
          std::atomic_thread_fence(std::memory_order_seq_cst);
          temp &= array2[array1[x] * 512];
     }
}
```

Secure Speculation Property



Evaluation

- Paul Kocher's list of 15 bounds check bypass examples
- Bounded model checking for exploit finding (5 steps)
- Inductive model checking for verification (1 step)

Example	Ex1	Ex5	Ex7	Ex8	Ex10	Ex11	Ex15	Fig. 3c	NI
BMC	6.6 secs	9.0 secs	10.2 secs	5.7 sec	9.6 secs	6.4 secs	5.8 secs	6.6 secs	12.9 secs
Induction	5.0 secs	5.0 secs	5.7 secs	4.6 secs	5.8 secs	5.9 secs	4.8 secs	4.8 secs	5.4 secs

Conclusion

- Recently advances in side-channel attacks warrants formal verification of microarchitectural models and trusted systems
- The Trusted Abstract Platform is an abstraction of enclave platforms and enables formal reasoning about secure remote execution
- A verification methodology was introduced to verify the absence of transient execution attacks on programs running on a microarchitecture
- Future work
  - Enabling automated and agile verification for enclave platform and microarchitectural processor designs using these existing methodologies
  - Showing refinement of TAP to the implementation of Keystone