T-SGX: Eradicating Controlled-Channel Attacks Against Enclave Programs

Ming-Wei Shih Sangho Lee Taesoo Kim Marcus Peinado

Georgia Institute of Technology *Microsoft Research*

The cloud is growing 7 times faster than the rest of IT

The latest IDC forecast says public cloud spending will grow almost 25% this year, topping $122 billion. And the growth keeps up through 2020.

Network World | FEB 21, 2017 12:25 PM PT

62 Percent of Companies Store Sensitive Customer Data in the Public Cloud

And almost 40 percent of cloud services are commissioned without the involvement of IT, a recent survey found.

By **Jeff Goldman** | Posted February 21, 2017

Share

NEWS

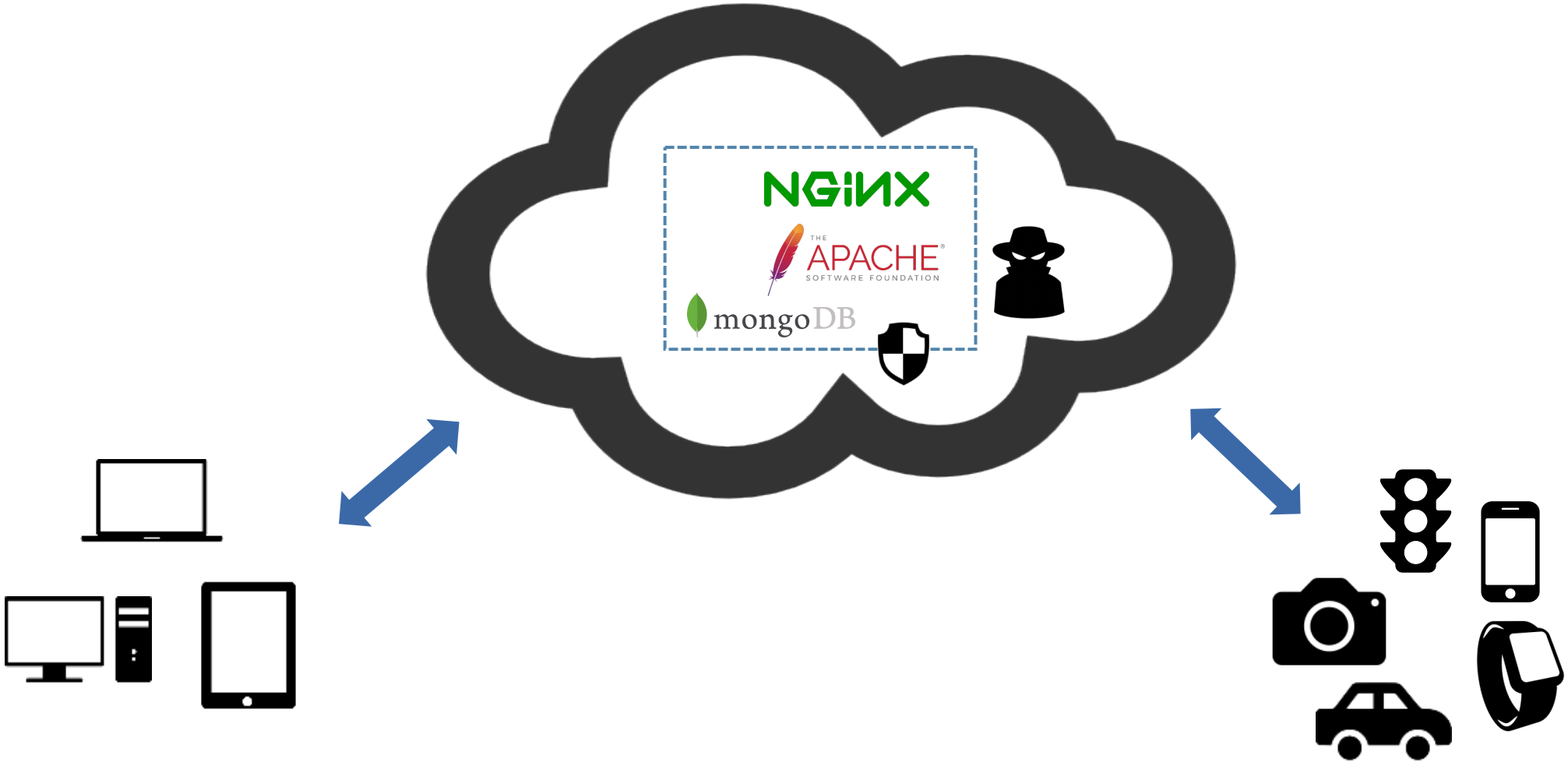
IT leaders say it's hard to keep the cloud safe

Shadow IT causing cloud trouble by illicitly working behind the scenes

By Sharon Gaudin | Follow

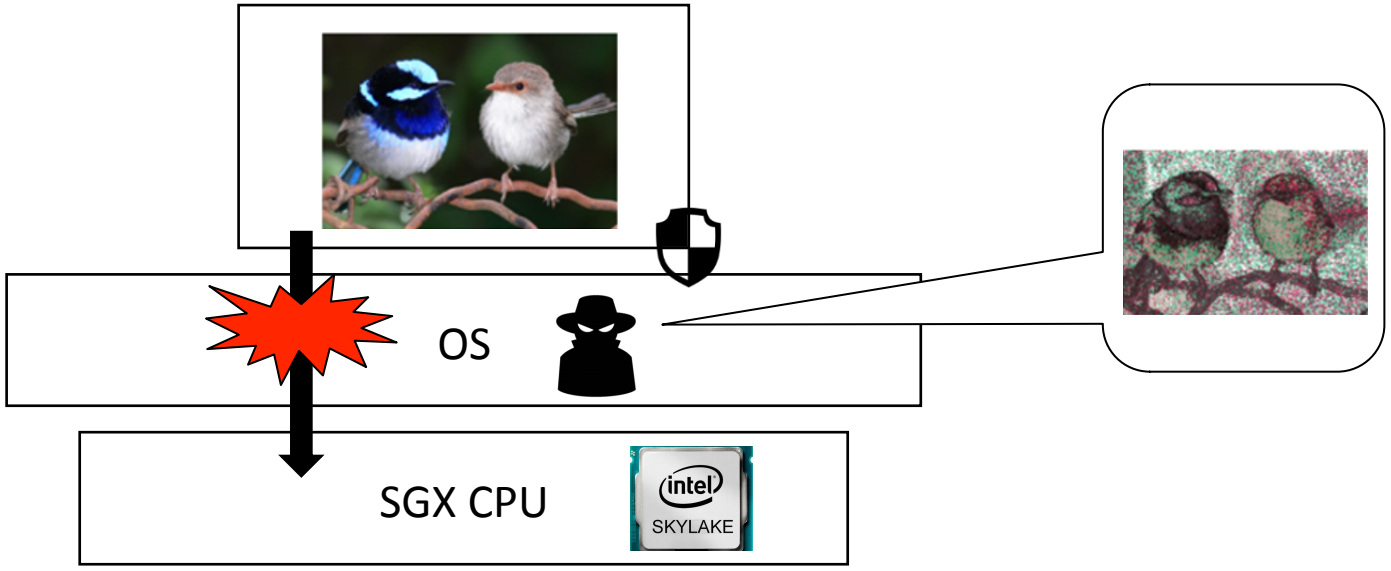
Senior Writer, Computerworld | FEB 15, 2017 12:17 PM PT

Intel SGX aims to secure users' code and data in the cloud



Controlled-channel attack [Oakland 2015] raises concerns

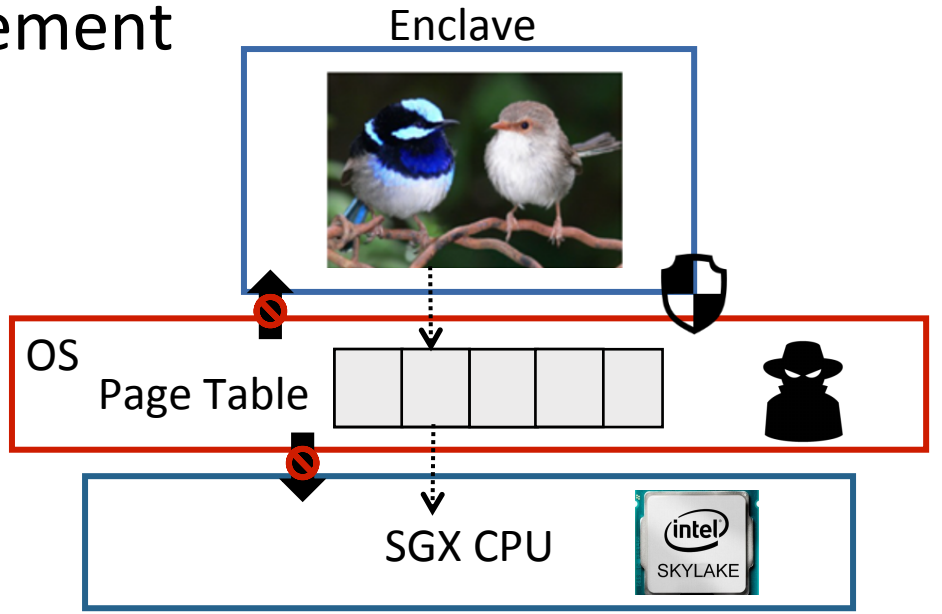
- An accurate side-channel attack that extracts the SGX-protected data
- Compromise the security guarantees of SGX



How the attack works (1/3)

- Intel SGX protects *enclaves* against an untrusted OS

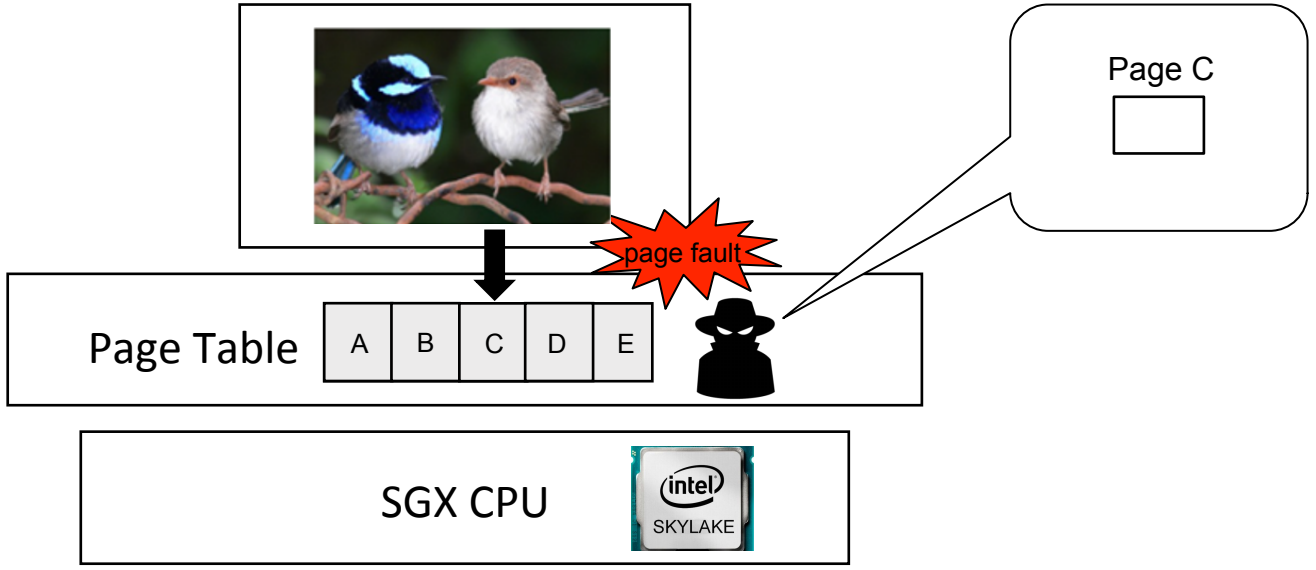
- SGX still relies on the OS for resource management (e.g., memory mapping)



How the attack works (2/3)

- Attacker fully controls the OS

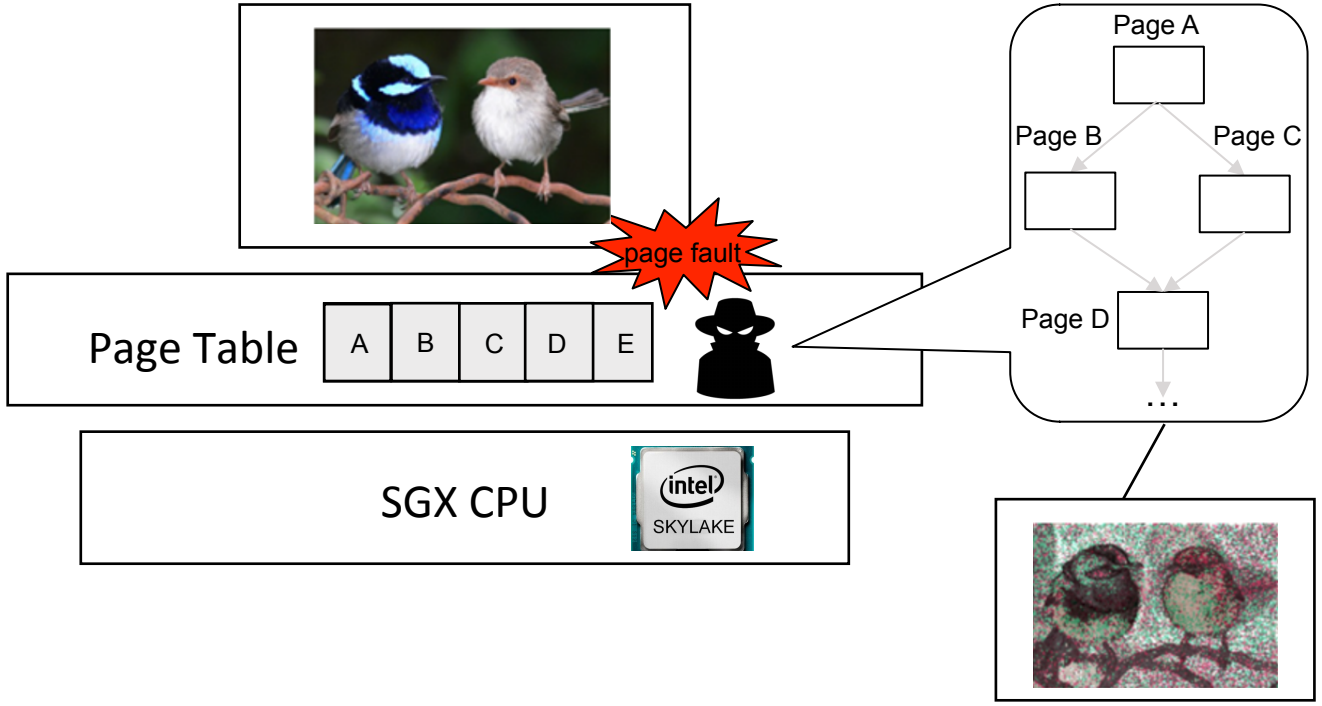
- Page-fault side channel
  - Step 1: Unmap a page
  - Step 2: Enclave accesses the page
  - Step 3: Observe a page fault



How the attack works (3/3)

- If the program's memory accesses depend on a ***secret***, then this ***secret*** is being leaked

- Attack steps
  - Offline analysis
  - Obtain page-fault sequence
  - Infer the secret

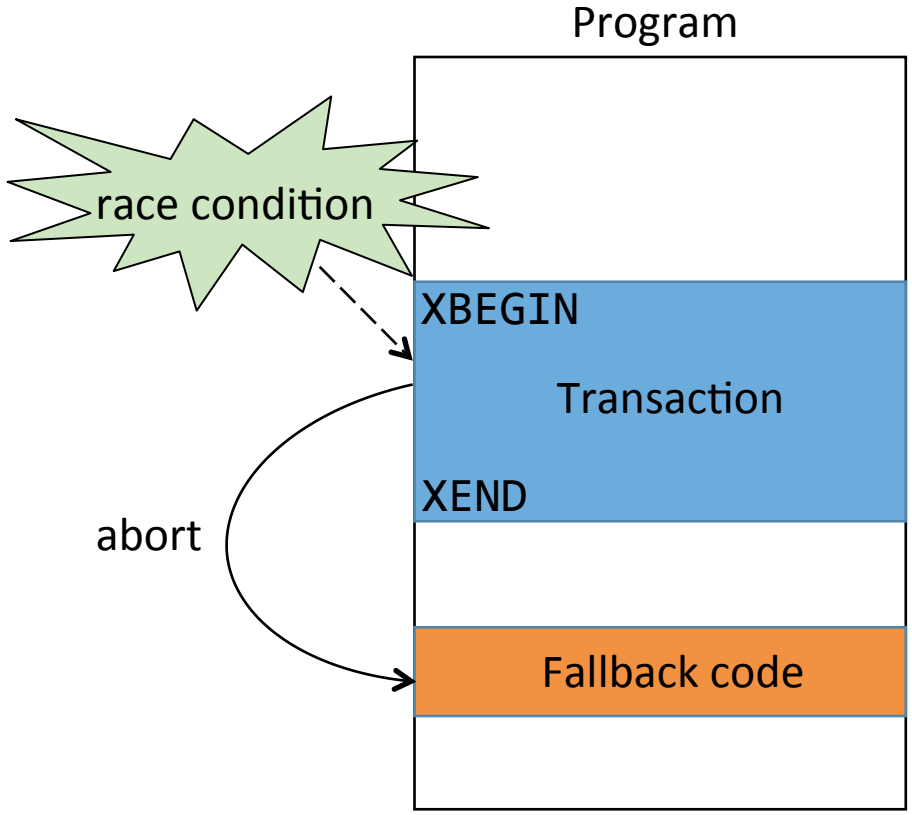


T-SGX Goals

- Prevent the controlled-channel attack
- The design should be practical
  - No hardware modification
  - Reasonable performance
  - Minimal developer effort (no need for program rewritten)

Intel TSX

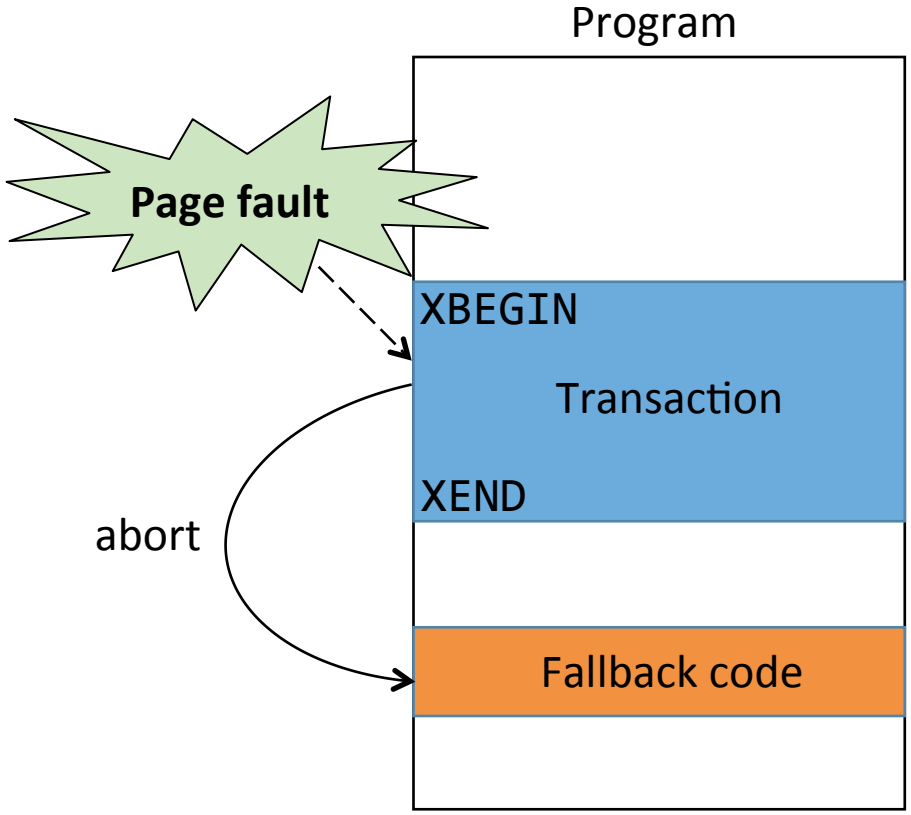
- CPU extension present in all recent Intel CPUs (since 2013)
- Supports hardware transactional memory
- Race conditions cause transaction abort
- An abort triggers fallback execution
  - Rolls back all changes
  - Control transfers to the fallback point



Idea: Intel TSX also suppresses page faults

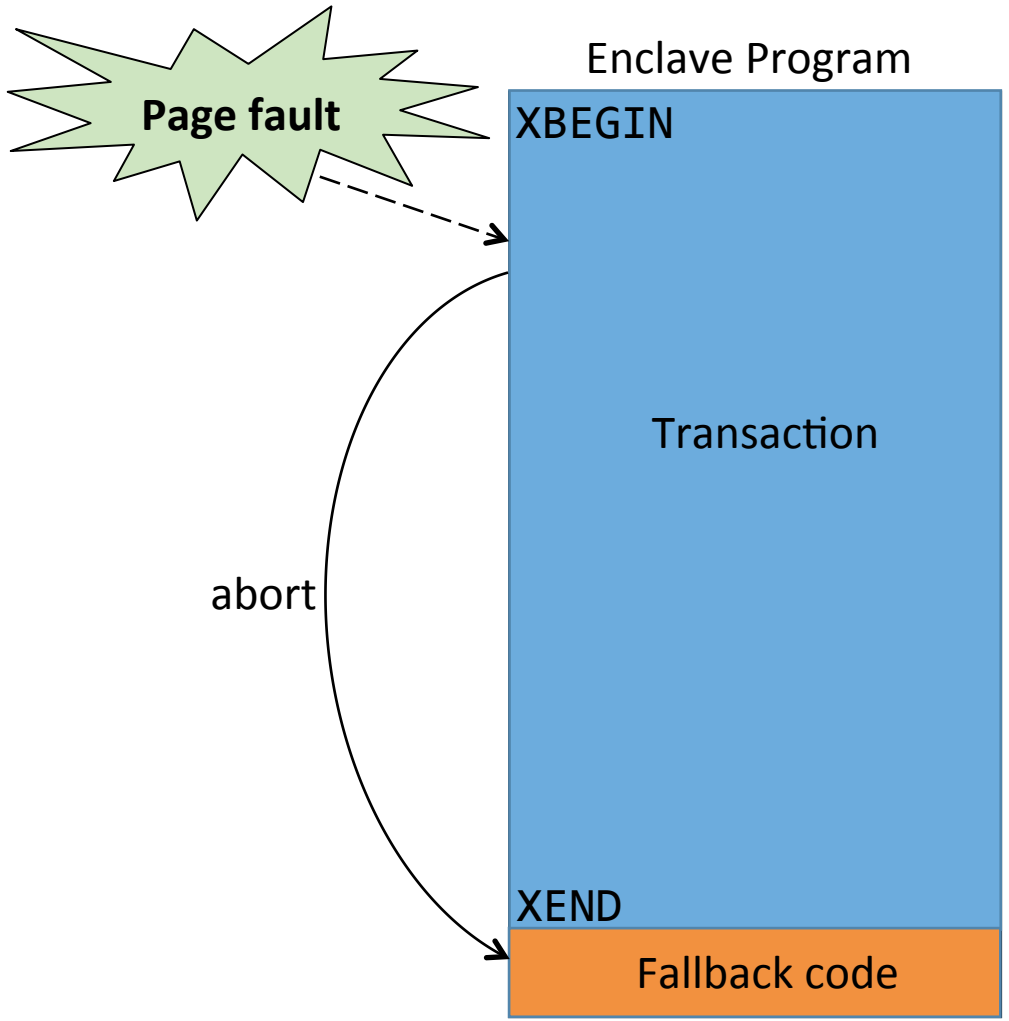
- CPU does not deliver page faults to the OS
- Instead, it aborts the transaction and invokes the fallback code

- OS ***cannot observe*** the page fault inside a transaction



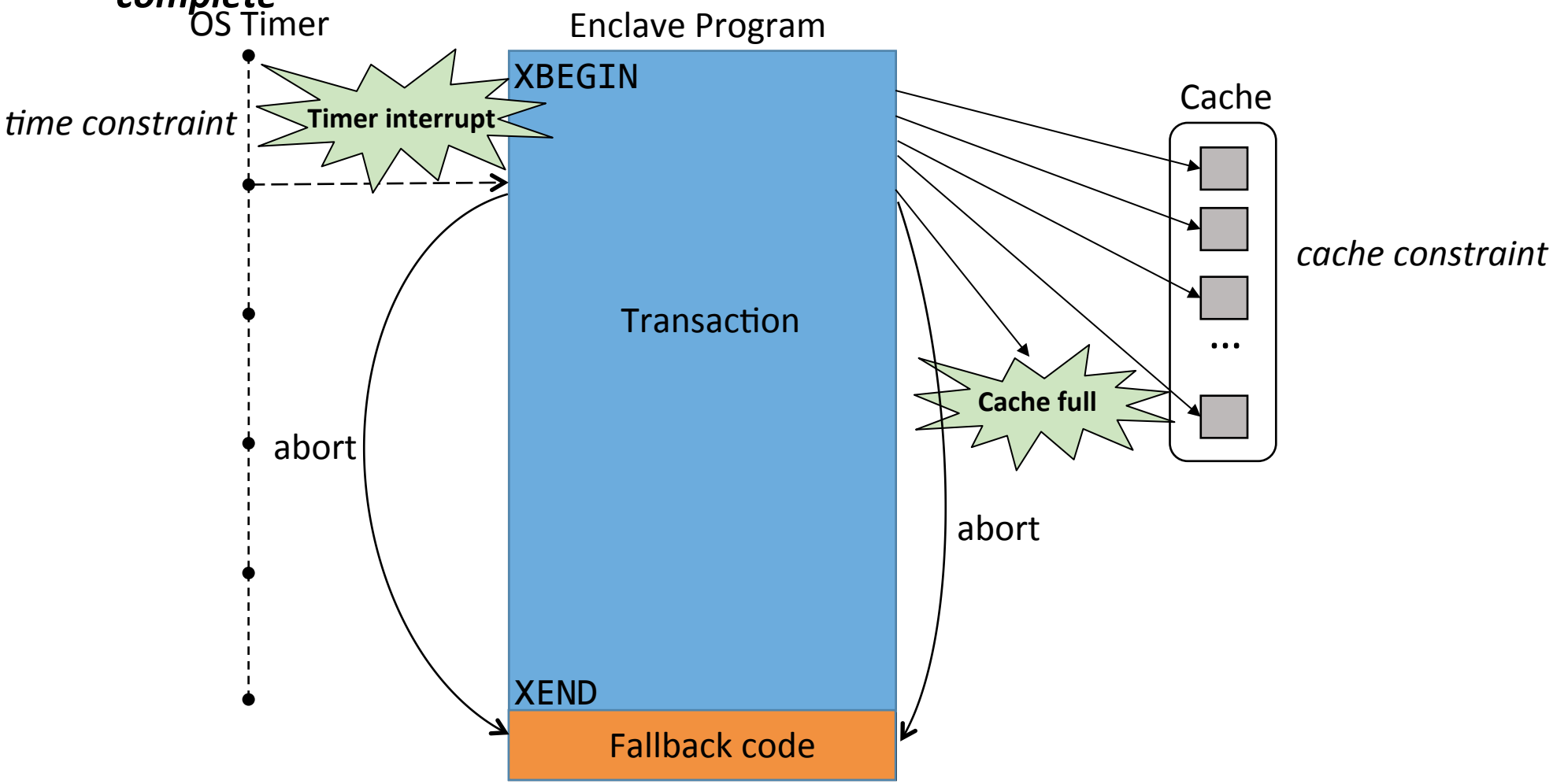
The strawman design

- Make the whole enclave as a transaction
- Enable the self-detection to page faults inside the enclave

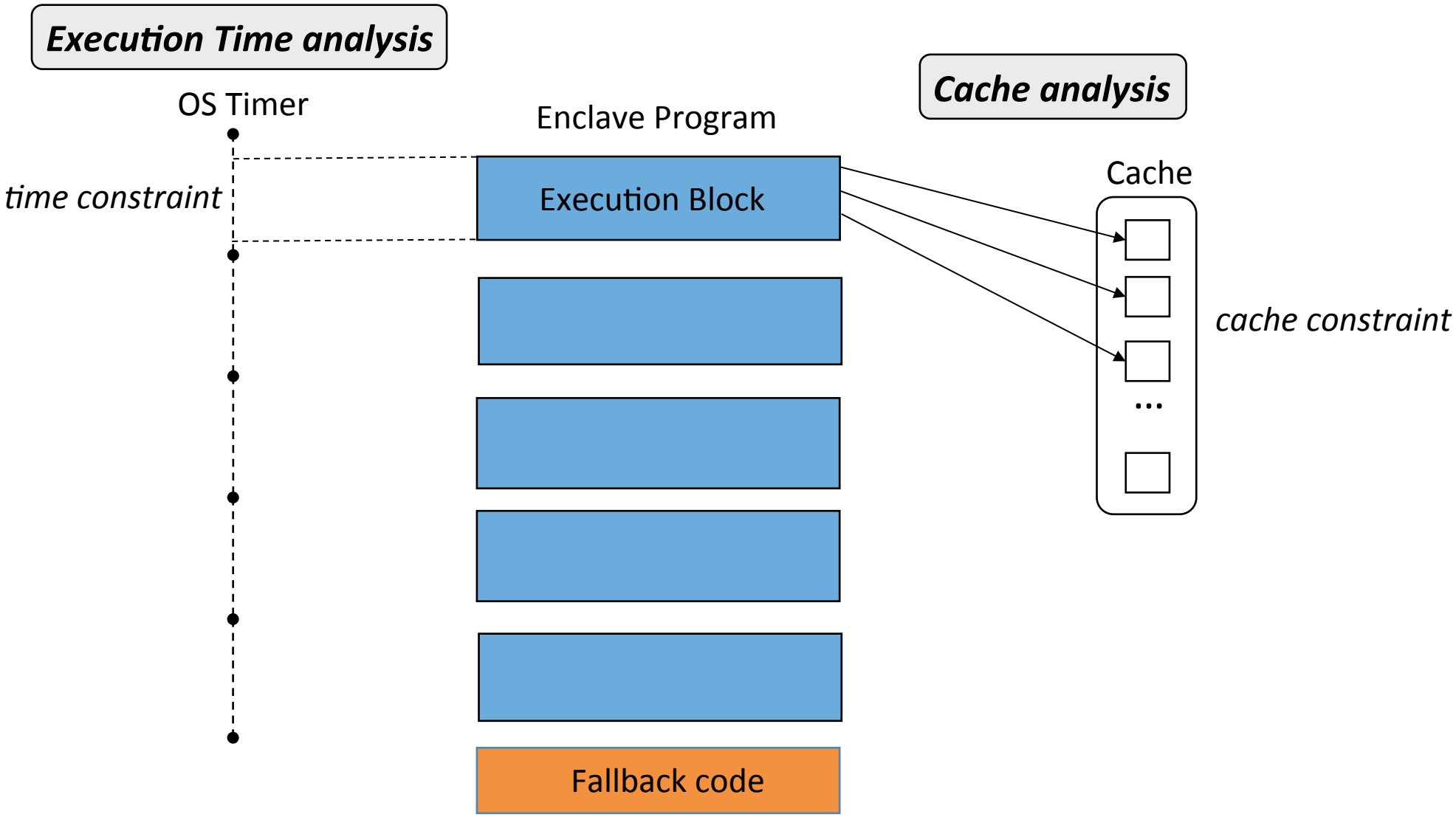


Challenges

Single transaction cannot be too large, otherwise it will never complete



Solution: Break a program into execution blocks



Optimization: merging tiny blocks (1/2)

- ***Problem***: Setting up transaction comes with a fixed cost (~200 cycles)

- If continuous blocks satisfy the cache and time constraints, we merge them
  - Loops
  - If-else statement
  - Functions

Optimization: merging tiny blocks (2/2)

- Example: Loop optimization

```
for (i = 1; i < 1000; i++) {
    XBEGIN
        …
    XEND
}
```



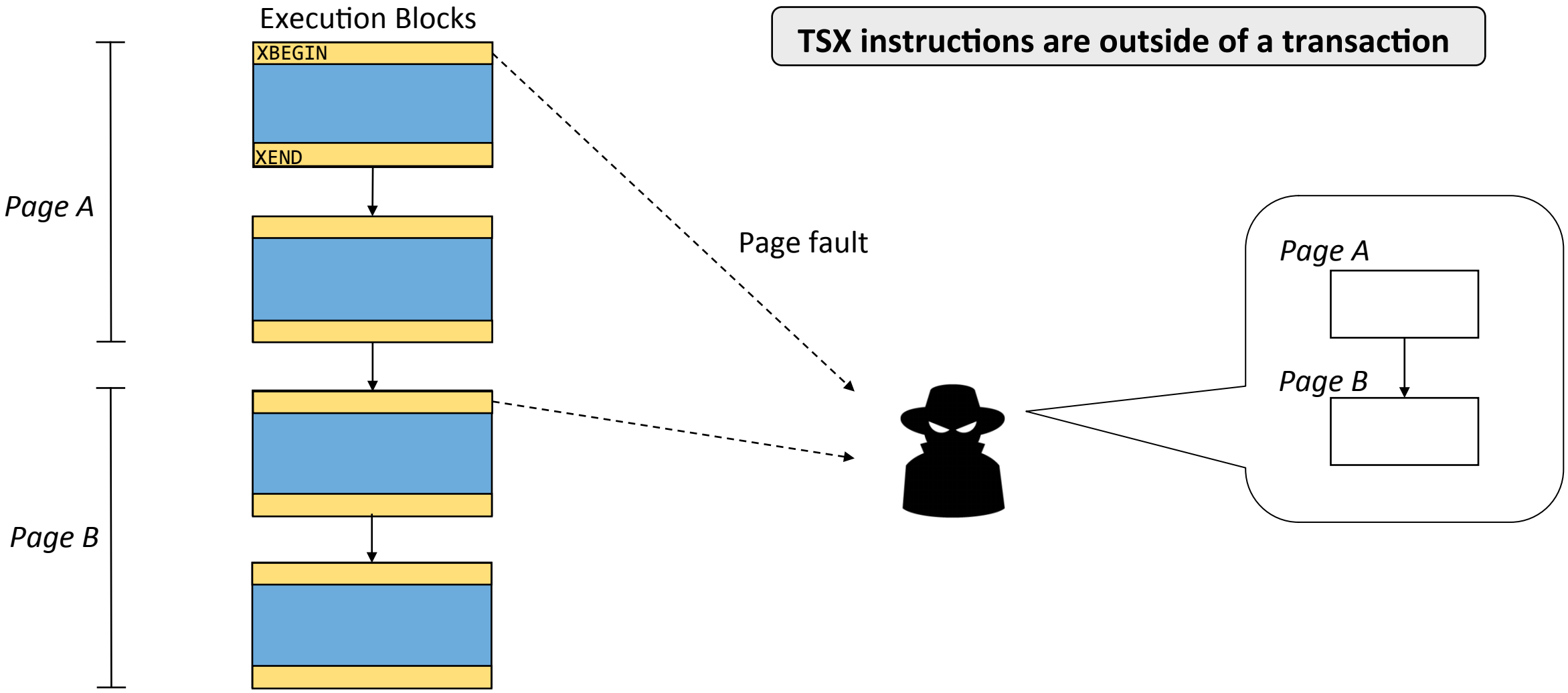
```
XBEGIN
for (i = 1; i < 1000; i++) {
        …
}
XEND
```

Requires 1000 transactions

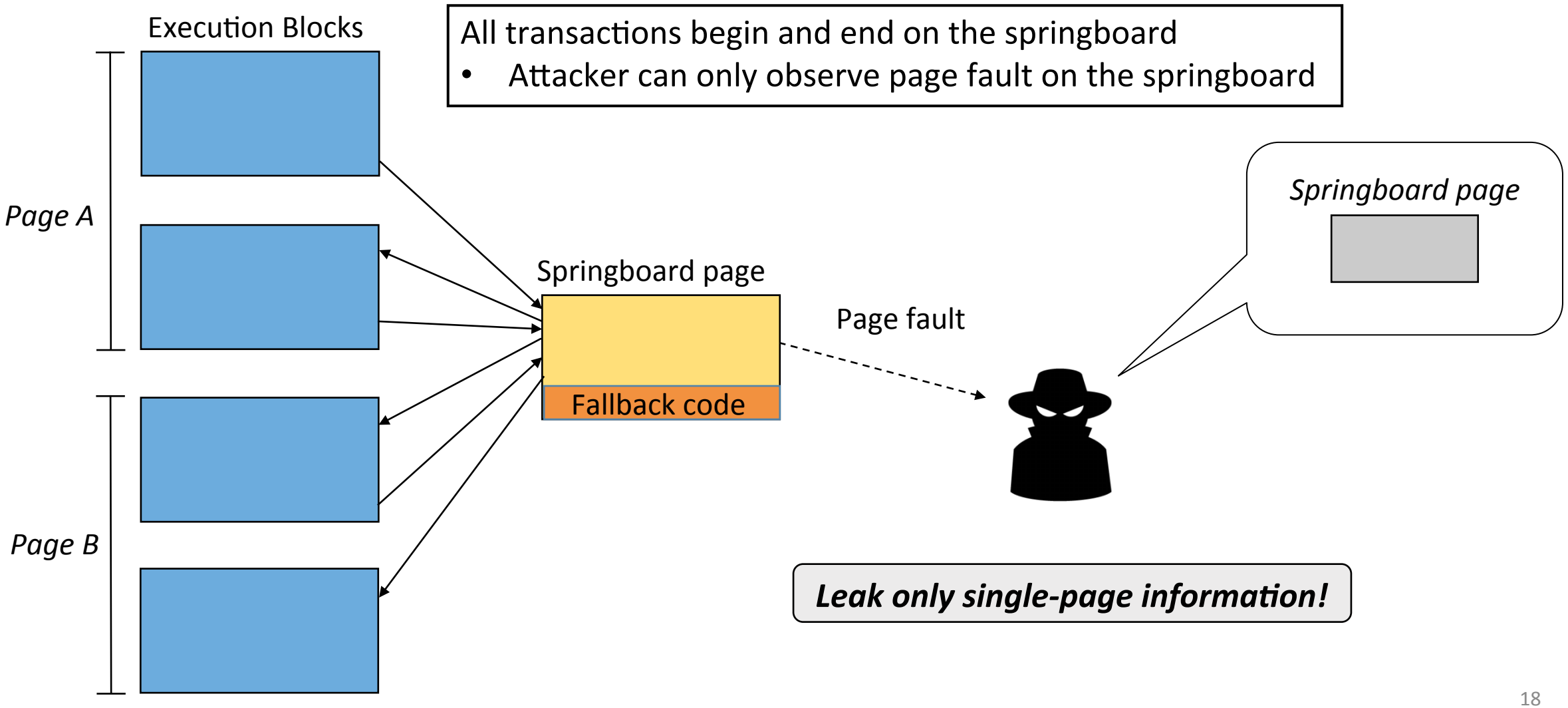
Requires only 1 transaction!

Conservative static analysis
- Only optimize when it's safe

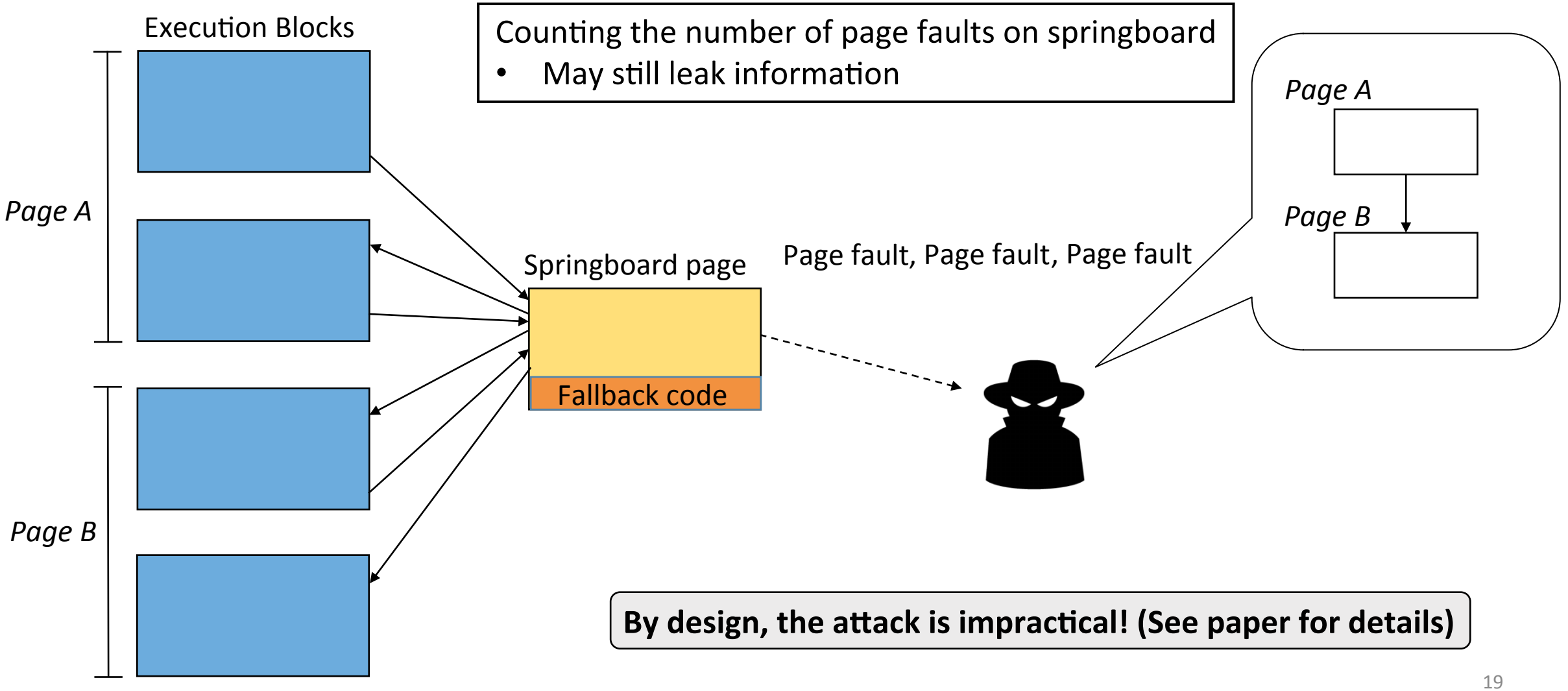
This design still leaks information



Solution: Springboard design



Springboard design also prevents advanced attacks



Implementation: T-SGX

- Based on the LLVM compiler
  - Mostly modifying LLVM backend
  - 4,100 line of code
  - Fully automated program transformation

Evaluation

- How general is the T-SGX approach?
- How much overhead does a transformed program have?

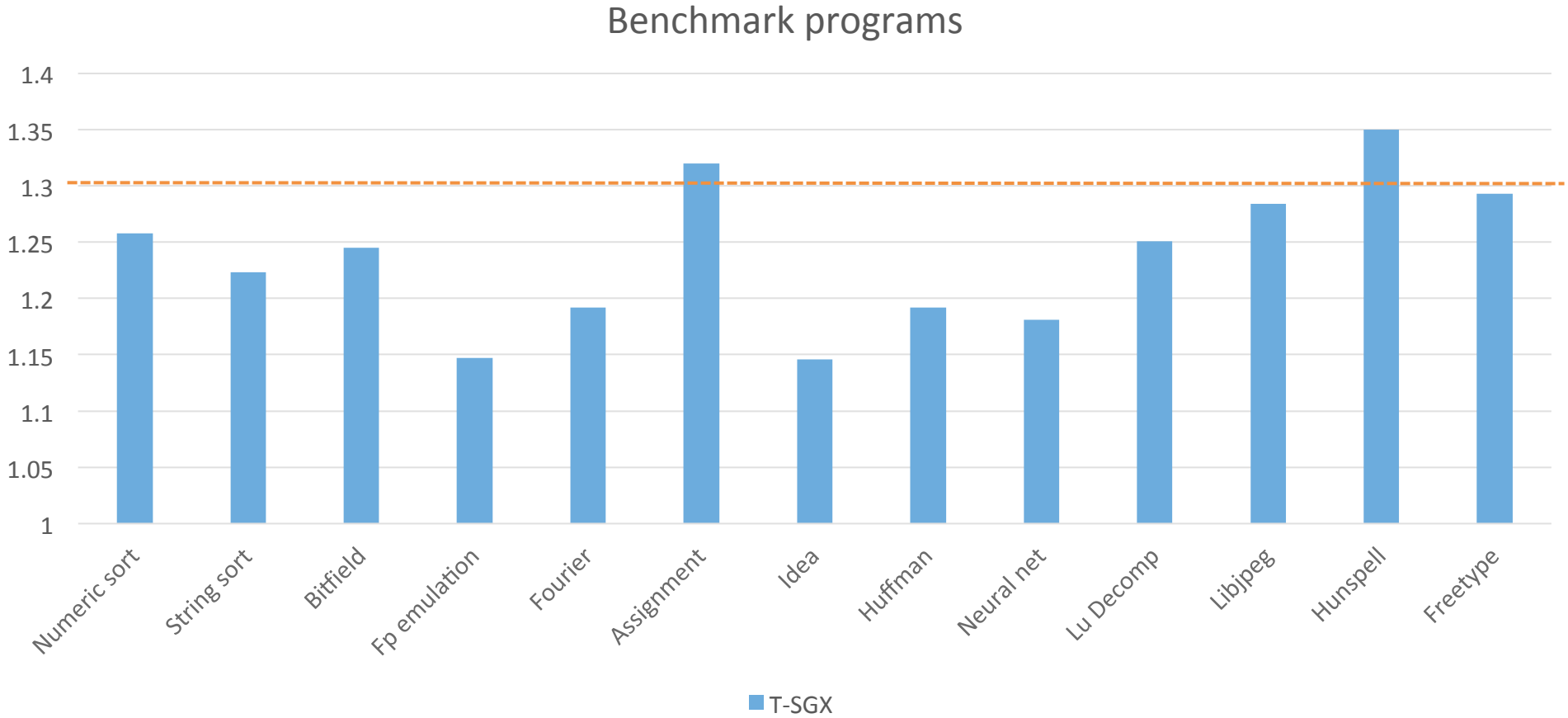
T-SGX works for general C/C++ programs

- 0 lines of source code change
- Fully-automated compiling chain

Application	Line of Code
Numeric sort	211
String sort	521
Bitfield	225
Fp emulation	1,396
Fourier	235
Assignment	490
Idea	353
Huffman	448
Neural net	746
Lu decomposition	441
Libjpeg	34,763
Hunspell	24,794
FreeType	135,528

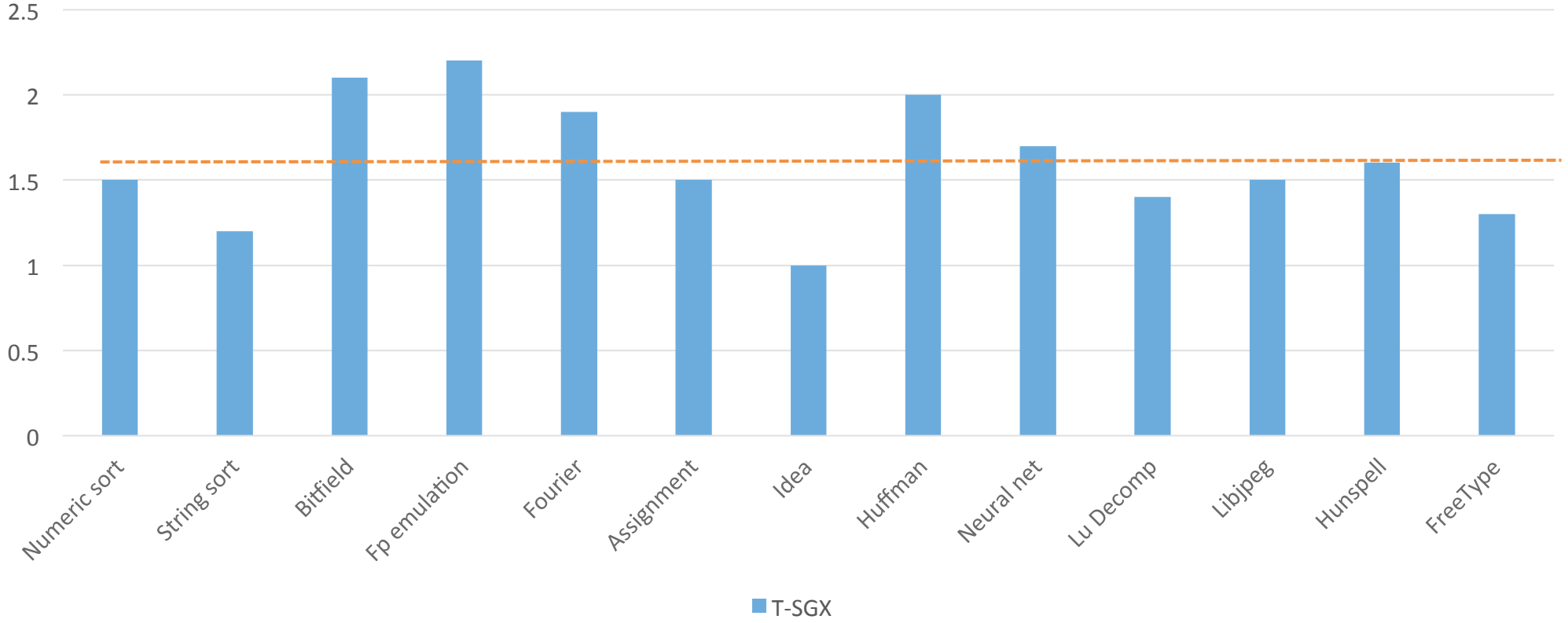
T-SGX incurs reasonable overhead

- Average 30% memory overhead
  - Additional instructions for each execution block

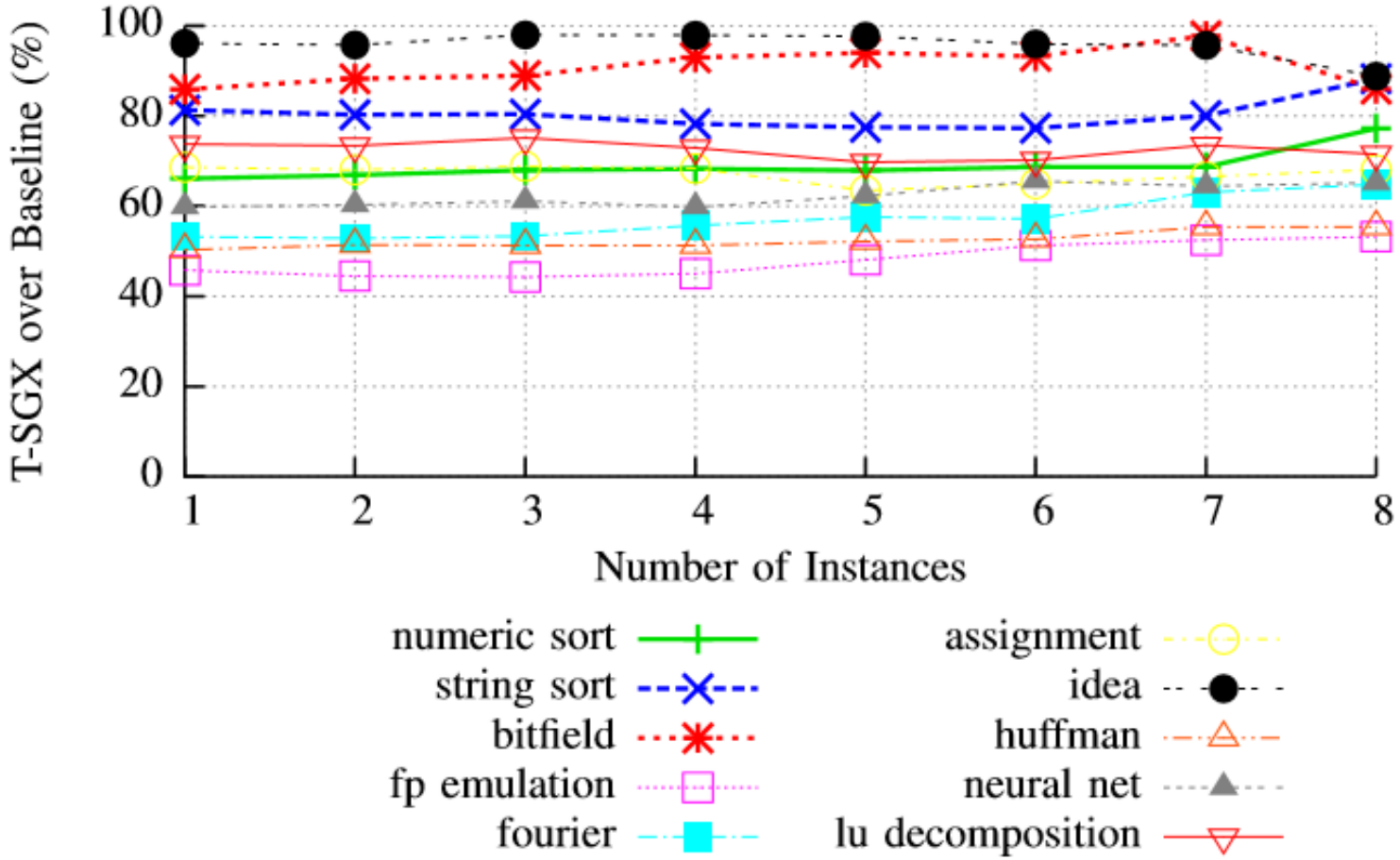


T-SGX incurs reasonable overhead

- Average 50% runtime overhead (geometric mean)
  - Largely depends on number of loop iterations that repeatedly start a transaction



Consistent runtime overhead on concurrent execution



Conclusion

- We proposed and implemented T-SGX, which effectively protects enclaves against the controlled-channel attack.

- T-SGX
  - Requires no hardware modification
  - Incurs reasonable runtime overhead and still has potential to improve (e.g., using more advanced program analysis or performance profiling)
  - Automatically transforms a program without the need for manual effort

Q&A