Designing a Provenance Analysis for SGX Enclaves

Flavio Toffalini, EPFL
Mathias Payer, EPFL
Jianying Zhou, SUTD
Lorenzo Cavallaro, UCL

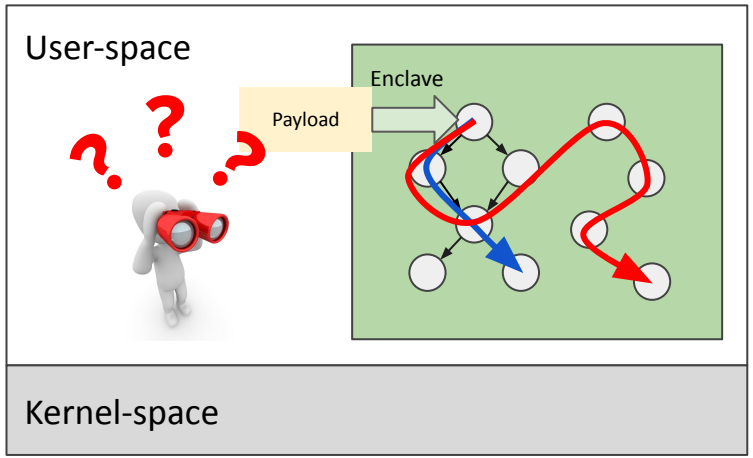
EPFL UCL SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN hexhive

Problem Description: Memory Corruptions in SGX

SGX protects the execution of software in an enclave (blue execution)

But Enclave software may be vulnerable to memory errors

-> Chain code gadgets to execute arbitrary malicious computations (red execution)



Challenges

What do we need? A **provenance analysis**!

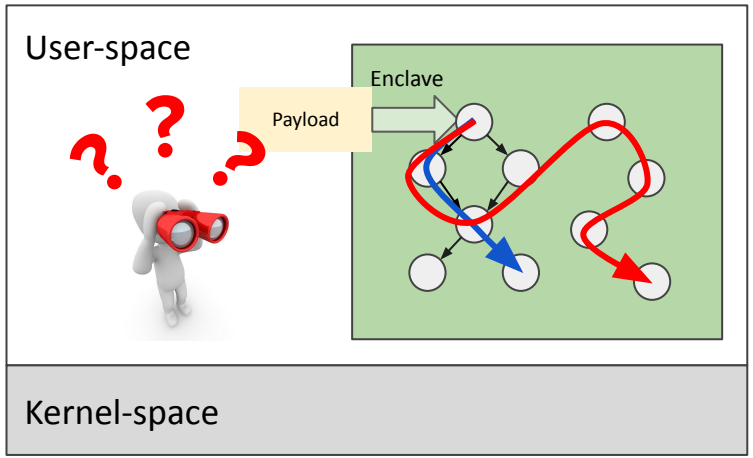
But SGX does not allow inspection :(

Challenges:

1) Attack-resistant tracing

2) Secure streaming

3) A model to recognize intrusion

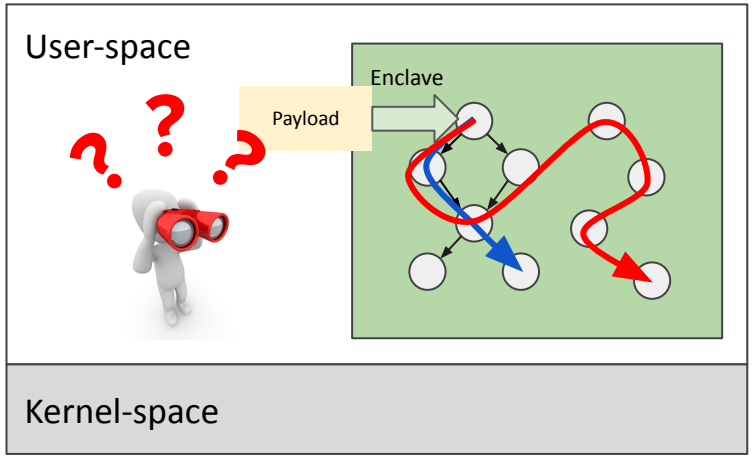


Contribution

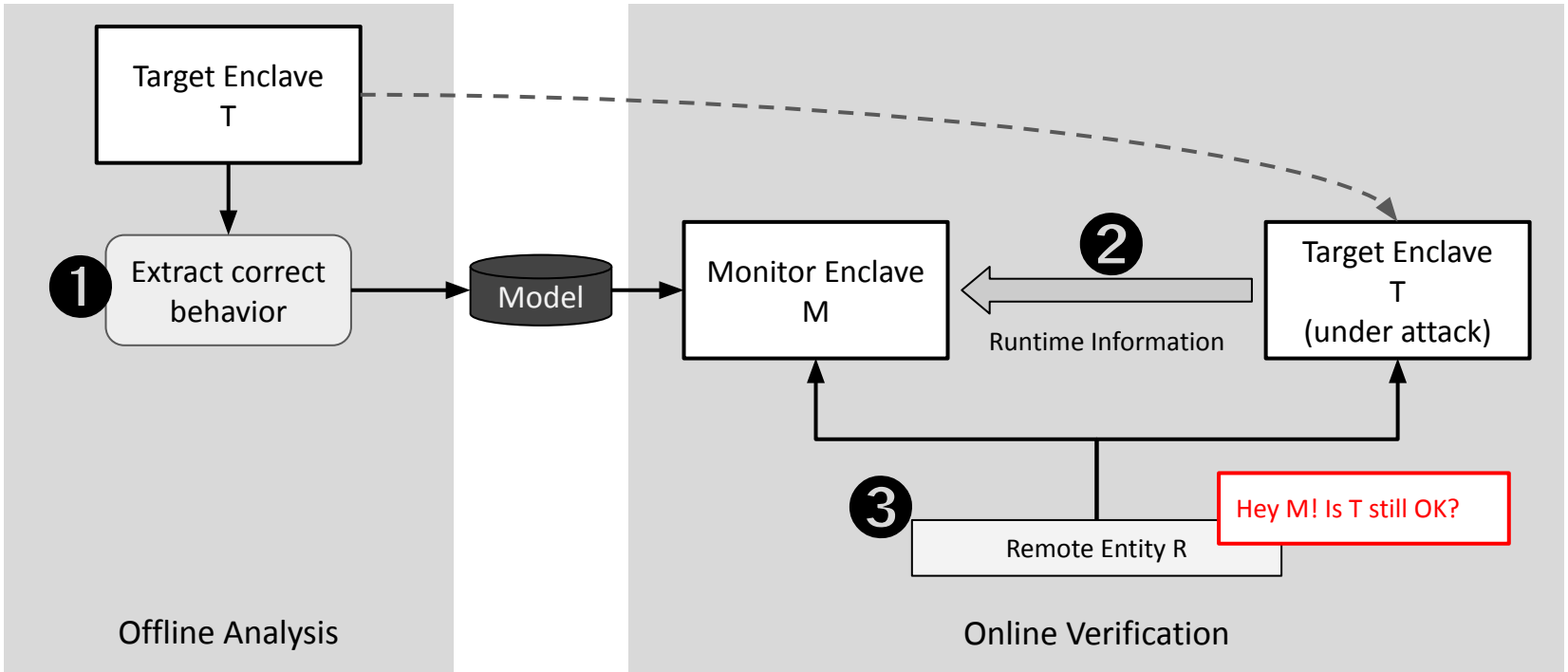
SgxMonitor: a provenance analysis for SGX!

1) Something to trace the enclave (securely)

2) A model to identify the attack



Design

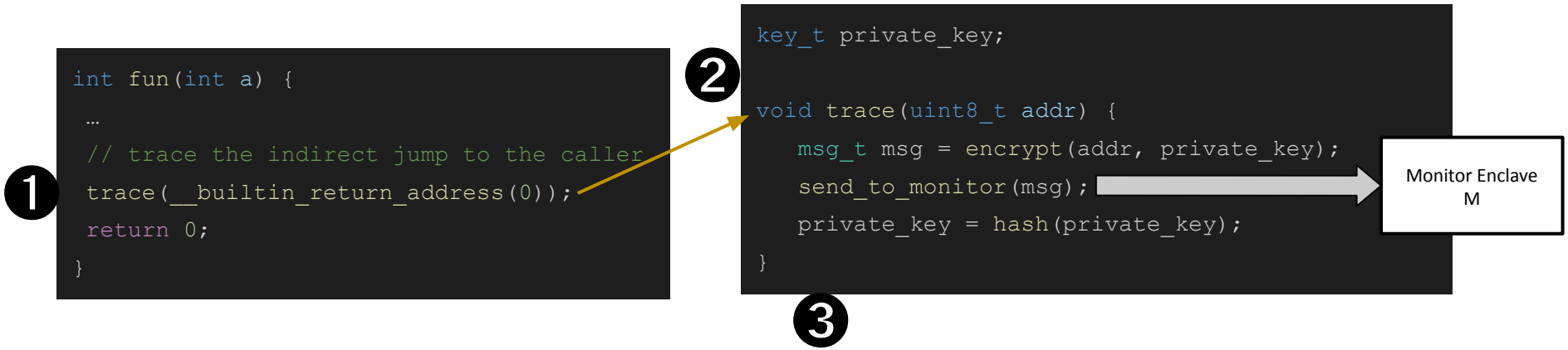

Target Enclave
T
1
Extract correct
behavior
Model
Monitor Enclave
M
2
Runtime Information
Target Enclave
T
(under attack)
3
Hey M! Is T still OK?
Remote Entity R
Offline Analysis
Online Verification

Design: Tracing

Attack-resistant tracing

Gist: every trace() sends an encrypted msg AND produces a new private_key

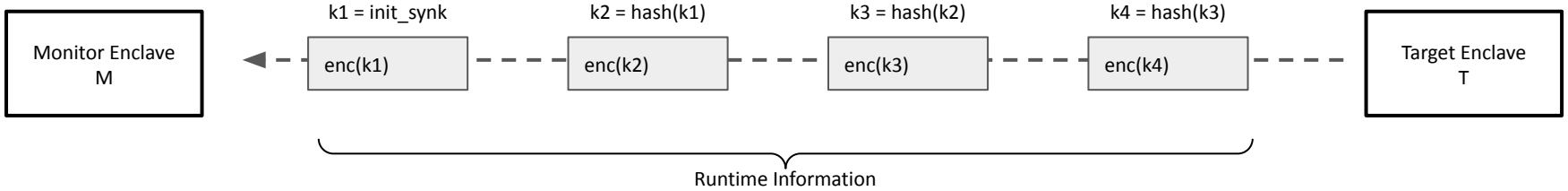
If an adversary leaks a key, it cannot be used to retrieve previous keys



Design: Streaming

Secure streaming

Gist: the messages are chained, dropping one reveals an attack.
Messages have same size, so no information of their content.

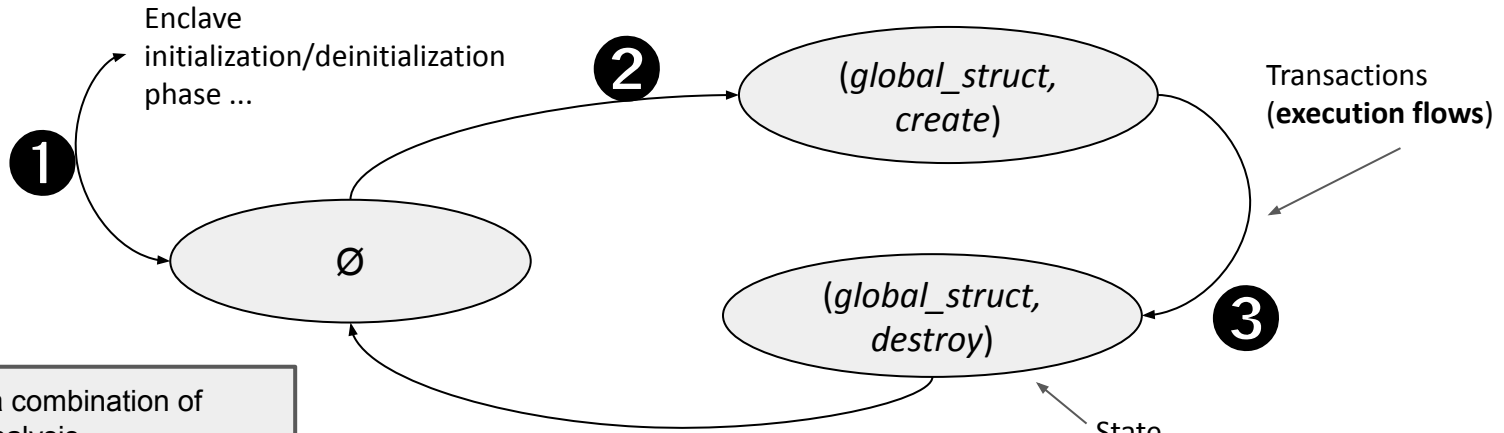


In the paper, a detailed security discussion

Model

Enclaves are stateful -> they use global variables/structs

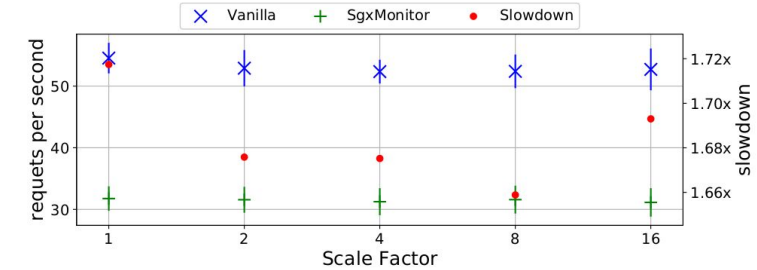
Assuming we know what global structures I need to protect

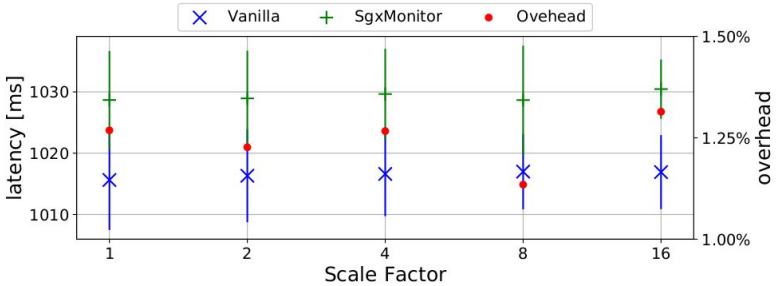


Extracted with a combination of symex+static analysis.
Static analysis used only as fallback if symex reaches timeout.

Evaluation: Overhead

Deployed over StealthDB (PostgreSQL plugin w/ SGX). Not that bad…





Acceptable overhead

Evaluation: Security

- Tried against SnakeGX[1], an SGX malware -> **stopped!**

- Tested mimicry attacks and shadow stack integrity -> **stopped!**

False positive or false negative observed: **none**

[1] SnakeGX: a sneaky attack against SGX Enclaves (ACNS 2021)

Takeaway!

- Runtime tracing mechanism for SGX enclaves
  - Without introducing new attacks surface

- Model SGX enclaves as a FSM (including global states)
  - Using symex+static to extract the model

- Evaluation
  - Macrobenchmarks show limited overhead
  - Model identifies and describes the attacks (no false positives observed)

https://github.com/tregua87/sgxmonitor-artifact

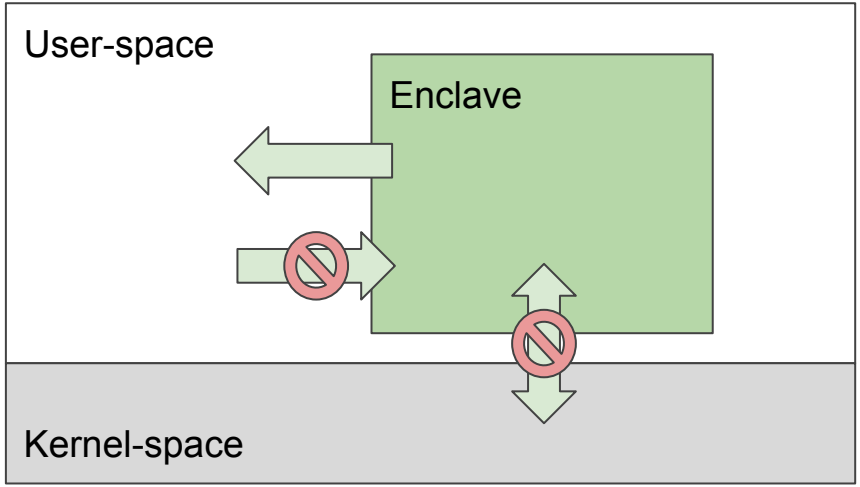
EPFL UCL SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN hexhive

backup…

SGX - Background

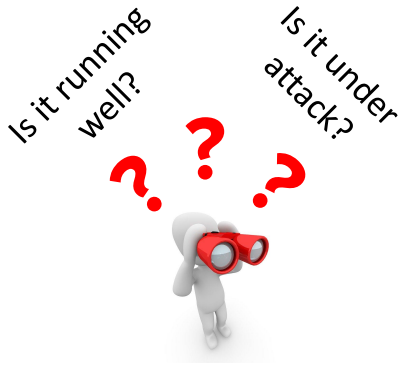
Intel Software Guard eXtention (SGX)

- Enclaves: isolated memory regions in user-space
- Enclaves cannot interact with ring-0 software (i.e., no syscall)
- Enclaves can write/read in user-space
- User- and kernel-space cannot write/read the enclave space



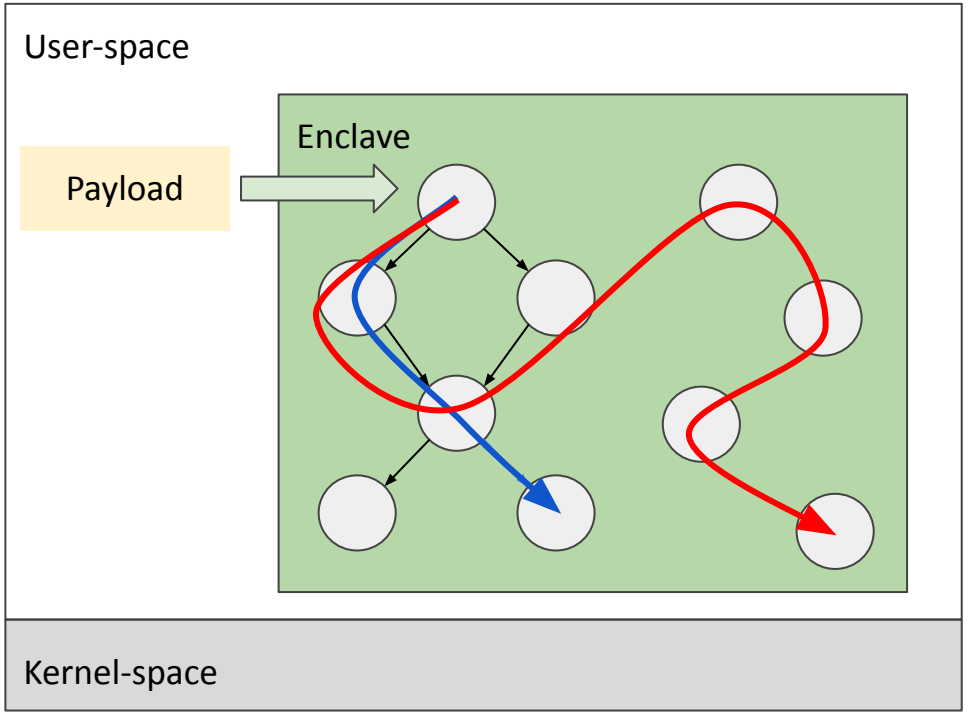
How is this enforced?
CPU/MMU/Microcode checks
OS-independent design

Problem description - memory corruptions in SGX



Correct execution

Hijacked execution



EPFL

hexhive

Tracing: Challenges

I want **something like Intel PT!**

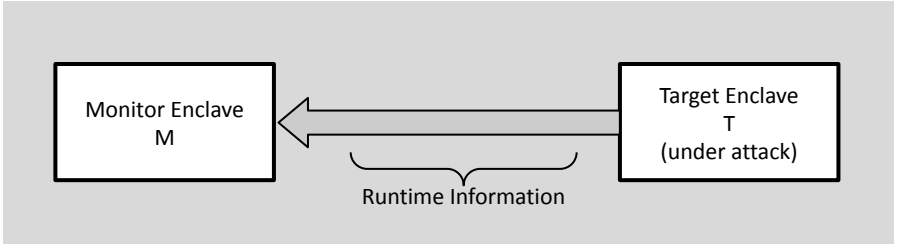
But SGX does not allow inspection :(

Challenges:

1) Attack-resistant tracing

2) Secure streaming

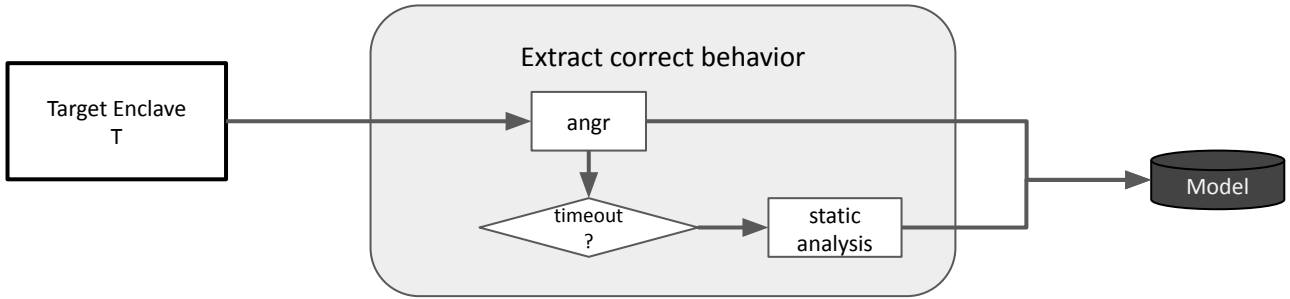
3) Not amplify side channels



Model Extraction

How do I extract the model?

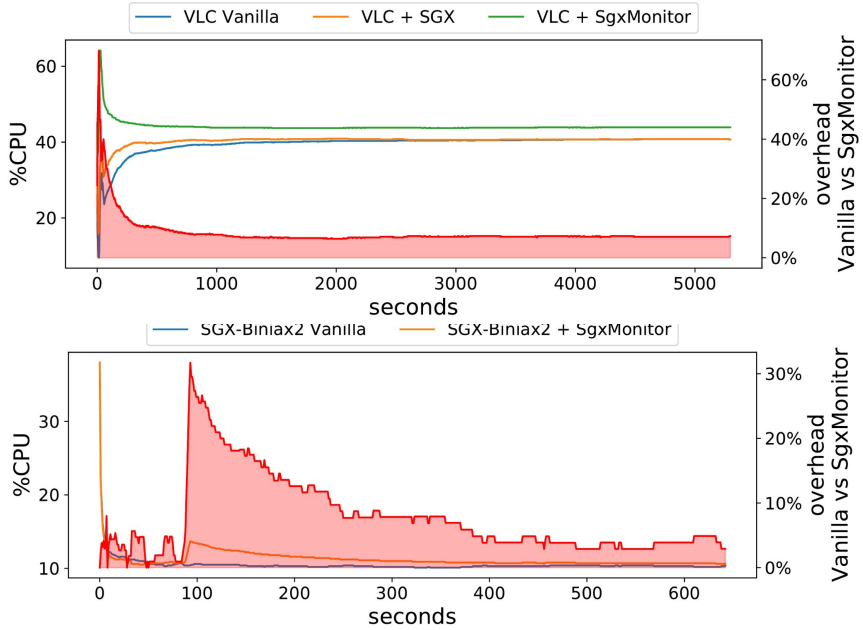
Gist: we extract CFG from every function by using static analysis and symbolic execution



More model/analysis details in the paper

Evaluation - overhead

Deployed over VLC (manual porting) and SGX-Biniax (an SGX game). Not that bad…



Macro-benchmarks show a plateau, thus not affecting final user experience

Evaluation - model precision

Use Case	# functions	% CFG explored	# functions static
Contact	71	96.4%	1
libdvdcss	56	91.4%	9
StealthDB	44	96.6%	0
SGX-Biniax2	49	91.6%	4
Unit-test	17	94.0%	0

Symex explores the majority of the functions
We fallback to static analysis only for few cases

EPFL

hexhive

Design: Is it Secure?

Does SgxMonitor amplify side channels?

We conduct this analysis.

We recall:
(i) all messages have same size, therefore the size does reveal
(ii) the target enclave changes its key for each message transmitted, thus leaking keys is useless