ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE | ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL | ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

USENIX Security ’24 Artifact Appendix: Divide and Surrender: Exploiting Variable Division Instruction Timing in HQC Key Recovery Attacks

Robin Leander Schröder   Stefan Gast   Qian Guo

A Artifact Appendix

A.1 Abstract

The artifact contains the source-code necessary to run the divide and surrender attack on HQC. The figures in the paper can be reproduced using the data produced by the attack.

A.2 Description & Requirements

The attack includes both a simulated and a “real” mode, which uses actual side-channel measurements as obtained by the DIV-SMT side-channel. To perform the actual side-channel attack an AMD Zen2 processor (Ryzen 3000) is required. If a Zen2 processor is not available, we provide the required data to reproduce the figure in `data/smt.csv`. All scripts and code is written for Linux and is not portable to other platforms.

A.2.1 Security, privacy, and ethical concerns

We see little risks involved in running the attack. The performance of the system may be impacted, as all cores are used. The `collect_division_thoughput_data.sh` script disables SMT and then restores the previous state.

A.2.2 How to access

https://github.com/hqc-attack/divide-and-surrender/tree/2d2d66c99736674e964cf162c40e226e90637f71

A.2.3 Hardware dependencies

An AMD Zen2 CPU is required to perform an actual DIV-SMT side-channel attack. The simulated modes have no special hardware requirements.

A.2.4 Software dependencies

Linux & docker/podman

When docker/podman is not used please review the Dockerfile for detailed information on dependencies.

A.2.5 Benchmarks

No data-sets or similar are required. We provide the output of the attack runs in the data directory. These can be used to reproduce the figures, but can also be regenerated from scratch.

A.3 Set-up

Install docker or podman (podman may require additional configuration).

A.3.1 Installation

```
docker build -t das .
```

Note that building will take a while (20 minutes on a 300Mbit/s connection). The final image is roughly 10GB.

A.3.2 Basic Test

Run the docker image:

```
docker run -it --rm \
    -v "$(realpath data)":/app/data \
    -v "$(realpath figures)":/app/figures \
    das
```

Then test if the existing data can be plotted:

```
python visualize/plot.py
```

The script should produce the following figures:

- Figure 9: `figures/n_traces.pdf`
- Figure 10: `figures/timing_histogram.pdf`
- Figure 11: `figures/difference_of_means.pdf`
- Figure 12: `figures/oracle_calls.pdf`

These figures should closely match the ones in the paper and will later be used to verify our claims. Mainly Figure 12 is relevant for claim C1. Once figure generation completes successfully you can clear the data directory:

```
rm data/*
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *Our HQC attack method requires significantly fewer oracle calls than previous methods. This is proven by Experiment E1.*

(C2): *The DIV-SMT side-channel allows key-recovery in HQC with high success probability. This is proven by Experiment E2.*

A.4.2 Experiments

(E1): [HQC Attack Simulation Results] [15 human-minutes + 24 core-minutes + <1MB disk]: Simulates the attack for various oracle accuracies. For each oracle accuracy it records each attack run's metrics in a CSV in 'data/'. We simulated 1000 attacks per oracle accuracy, but this will take longer and is not strictly necessary to validate claim C1.

Preparation: Inside the `das` docker container (see A.3.1).

Execution: Run the following two commands:

```
OPTS="--num-attacks=100" ./collect_simulation_data.sh
python visualize/plot.py
```

Results: Figure 12 is produced and saved in `figures/oracle_calls.pdf`. The number of oracle calls should (roughly) match the reported numbers in the paper for each oracle accuracy. Note that the 0.515 oracle accuracy is removed from the script, as it will take up the vast majority of the computation time, and isn't necessary to validate the major claim C1.

(E2): [DIV-SMT powered side-channel attack] [15 human-minutes + 32 core-minutes + <1MB disk]: Run the attack using the SMT oracle. This requires an AMD Zen2 processor, possibly even an exact match (AMD Ryzen 7 3700X) as we did not pursue portability of the DIV-SMT oracle in our implementation.

Preparation: Inside the `das` docker container (see A.3.1).

Execution: We performed 1000 attacks, but this will take hours and is also not strictly necessary to validate claim C2.

```
RUST_LOG=debug cargo run --release -- \
    attack --smt --num-attacks=8 \
    --stats-file=data/smt.csv
```

To evaluate the results run:

```
julia --project=. attack_stats.jl
```

Results: The results printed by the `attack_stats.jl` script should roughly match the numbers reported in Section 5.2 "Real-World Attacks in an AMD Zen2 Platform" in the paper.

If the attack is not run on an AMD Zen2 processor or the processor does not match our targeted processor closely enough (AMD Ryzen 7 3700X) the attacks should either stall and repeatedly log "Exhausted all 30000 measurement slots." or the attacks will terminate but fail.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.