ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

A Artifact Appendix

A.1 Abstract

This artifact contains a reference implementation of the ppSAT protocol, as well as supporting benchmarks and helper scripts necessary for reproducing our experimental results. Our implementation uses the EMP-toolkit framework for the underlying secure computation. Specifically, the artifact contains an implementation of the protocol as a secure distributed program, compiling which produces a ppSAT solver binary. This binary is then used by our testing infrastructure, and would in production be distributed amongst the parties and invoked over their local, private formulas to execute the SAT solving decision procedure over their conjunction. The artifact also includes functionality to support running microbenchmarks on individual giant steps of the ppSAT protocol over random formulas, as well as our haplotype inference benchmarks (and supporting scripts). It further includes code to execute the ppSAT protocol 'in the clear' to allow evaluating the overhead of the secure computation.

The artifact is composed of C++ code supported by Python scripts, intended for execution on suitable x86-based hardware running Ubuntu 20.04 or a similar, modern desktop Linux distribution. The compilation and benchmarking are verified by the Github Action CI.

A.2 Artifact check-list (meta-information)

- **Algorithm:** the ppSAT solver algorithm, including the underlying oblivious stack
- **Program:** implementations of private ppSAT and its tests, as well as of ppSAT 'in the clear'
- **Compilation:** cmake and make
- **Data set:** HapMap dataset, publicly available at https://web.archive.org/web/20170706011547/http://www.stats.ox.ac.uk/~marchini/phaseoff.html
- **Run-time environment:** tested on Ubuntu 20.04, should work on all modern desktop Linux distributions
- **Hardware:** tested on Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz * 6 processor
- **Execution:** shell scripts, python scripts, x86 binary
- **Security, privacy, and ethical concerns:** the HapMap dataset is publicly released and widely used in genetic studies
- **Metrics:** running time
- **Output:** the running time and model (or UNSAT) for satisfiable formulas (or unsatisfiable formulas)
- **Experiments:** all experiments are verified by a Github Action composed of four items: `functionalities` (unit testing of our protocol components); `microevaluation` (benchmarking of our protocol components, produces figures in §5.1); `simulation_difference` (used to verify that our estimated time is close to wallclock running time, as mentioned in §5.2); and `benchmark` (reproduces Sections 5.2 and 5.3 – we only include one test because running all benchmarks exceeds 6 hours of running time)
- **How much disk space required (approximately)?:** 10 GB
- **How much time is needed to prepare workflow (approximately)?:** 2 hours
- **How much time is needed to complete experiments (approximately)?:** 4 hours for benchmarking our cryptographic protocols; 12 hours for obtaining the number of steps that our solver needs for Haplotype benchmarks
- **Publicly available (explicitly provide evolving version reference)?:** https://github.com/PP-FM/ppsat
- **Archived URL :** https://github.com/PP-FM/ppsat/releases/tag/v1.0.0

A.3 Description

A.3.1 How to access

https://github.com/PP-FM/ppsat

A.3.2 Hardware dependencies

A modern x86 CPU.

A.3.3 Software dependencies

cmake, emp-toolkit, gtest, openssl

A.3.4 Data sets

The HapMap dataset. Our repo includes the data that are used for our benchmarking.

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

We use a publicly released, widely-used dataset.

A.4 Installation

Installation can be easily done by following the instructions at https://github.com/PP-FM/ppsat#installation. Our Github Action scripts contain all steps to install our code on a clean Ubuntu machine.

A.5 Evaluation and expected results

The latest Github Action results, at the point of submission, can be found at https://github.com/PP-FM/ppsat/actions/runs/1894455944. The output of each subtask includes the running time and status of each test. They can be used to plot the figures presented in the paper.

In the paper, we made the following claims.

1. Our ppSAT solver can correctly and reasonably efficiently solve SAT formulas based on our newly designed heuristics, and all components scale well when the size of the formula increases (§5.1). This claim is tested in `artifact/functionalites` and `artifact/microevaluation`.

2. Our ppSAT solver can be used towards a real application of solving haplotype inference (§5.2). The accuracy of our timing estimation is tested in `artifact/simulation_difference`, while the rest is benchmarked in `artifact/benchmark`.

3. Our ppSAT solver still incurs a high overhead compared with a plaintext solver (§5.3). The results from §5.3 only require a plaintext SAT solver, for which we use Kissat.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220119.