Artifact details

This is the research artifact for the paper *SoK: The Constant Time Model* at WOOT 2026.

Download from this URL:

https://gitlab.com/platsec/boringssl-keyload-vuln/-/tree/bbb_woot

or:

```
git clone -b bbb_woot --single-branch https://gitlab.com/platsec/boringssl-keyload-vuln.git
```

The goal of this artifact is to reproduce Figure 1 (Left, Center) from the paper. You'll see visually and statistically separable distributions for BoringSSL, while the OpenSSL distributions visually overlap and are not statistically separable.

Google and I were unable to agree on terms for the code used to produce Figure 1 (Right) from the paper. So unfortunately that part of reproducibility is not included in this artifact.

Timing Attack on BoringSSL Private Key Loading

This repo demonstrates a timing side-channel vulnerability in BoringSSL's private key loading functionality. Specifically, the time it takes to load a private key depends on its effective bit length, leaking information that can potentially be exploited by remote attackers. The running example in this demo uses keys for NIST elliptic curve P-384, but it probably works for others, too.

install dependencies

FTR I did this on a Debian 12.12 box, with an Intel i5-7500T CPU. Yet, my gut says the vulnerability is OS and architecture independent.

```
sudo apt install build-essential cmake python3
```

I'm sure there are more dependencies; PRs welcome.

build BoringSSL

```
cd /path/to/boringssl-keyload-vuln
cd ..
git clone https://boringssl.googlesource.com/boringssl
cd boringssl/
git checkout 88d0c0f4772f3abe74f4f1012fe580fa85bab417
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Debug ..
make -j4
```

You may or may not need the `git checkout` part for the specific commit ID. I'm just providing it here for a discrete commit where I developed this PoC.

You might want to adjust the core count (4), depending on your build architecture.

build OpenSSL

```
cd /path/to/boringssl-keyload-vuln
cd ..
wget https://github.com/openssl/openssl/releases/download/openssl-3.6.0/openssl-3.6.0.tar.gz
tar xf openssl-3.6.0.tar.gz
cd openssl-3.6.0/
./config -d
make -j4
```

build PoC

You might need to adjust linking paths in `Makefile` depending on your build environment. Also, select one of these lines, depending on if you want to link against BoringSSL or OpenSSL.

```
LDFLAGS ?= -I../boringssl/include -L../boringssl/build -lcrypto
#LDFLAGS ?= -I../openssl-3.6.0/include -L../openssl-3.6.0/ -lcrypto
```

Then just build the harness.

```
cd /path/to/boringssl-keyload-vuln
make
```

collect data

First disable TurboBoost and set the frequency governor. (Technically not really needed, but it reduces the number of measurements required in this PoC.)

```
./freq.sh 1 performance
```

Then run the experiment.

For BoringSSL:

```
taskset -c 3 ./keydist pems > keydist_boringssl.dat
```

For OpenSSL:

```
taskset -c 3 ./keydist pems > keydist_openssl.dat
```

You might have to adjust which core to pin to (3), depending on your runtime architecture. Which core shouldn't particularly matter, but the goal is to pin it to a single core. Even that probably doesn't really matter, but MAH REPRODUCIBILITY.

Impatient? I feel you, dawg. If you're OK with noisier statistics, you can play with the repetition count. Fewer repetitions means less time for the experiment to run, at the cost of precision.

```
$ grep -n BBB_REPS keydist.c
15:#define BBB_REPS (1000)
```

After completing the experiment, your data (`keydist_*.dat`) should look something like this.

```
pems/priv_296_48.pem,9097673
pems/priv_296_48.pem,9106362
...
pems/priv_136_30.pem,9096688
pems/priv_136_30.pem,9092763
...
```

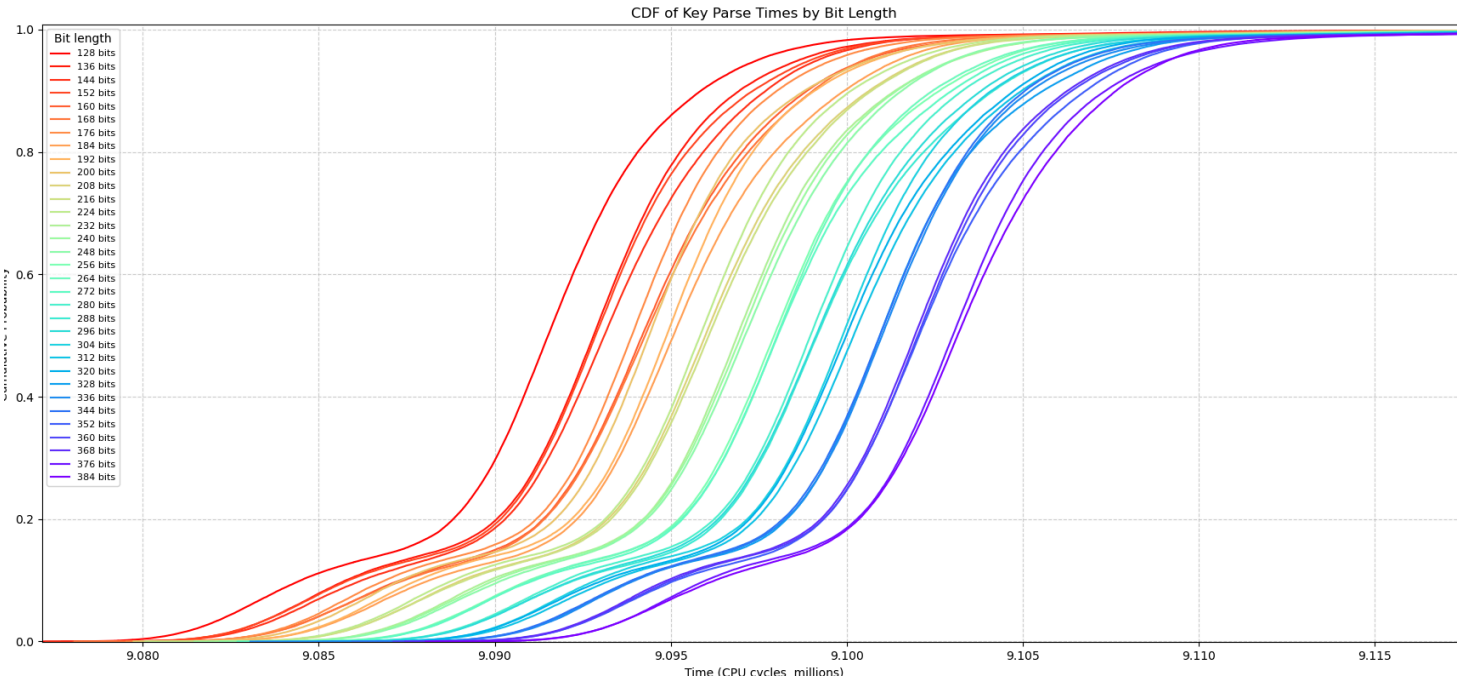
visualize data

The python script will probably barf at first, until you install whatever rando `python3` package dependencies it needs. (I didn't write `keydist.py` -- ChatGPT did.)

visualize data: BoringSSL

```
python3 keydist.py keydist_boringssl.dat
```

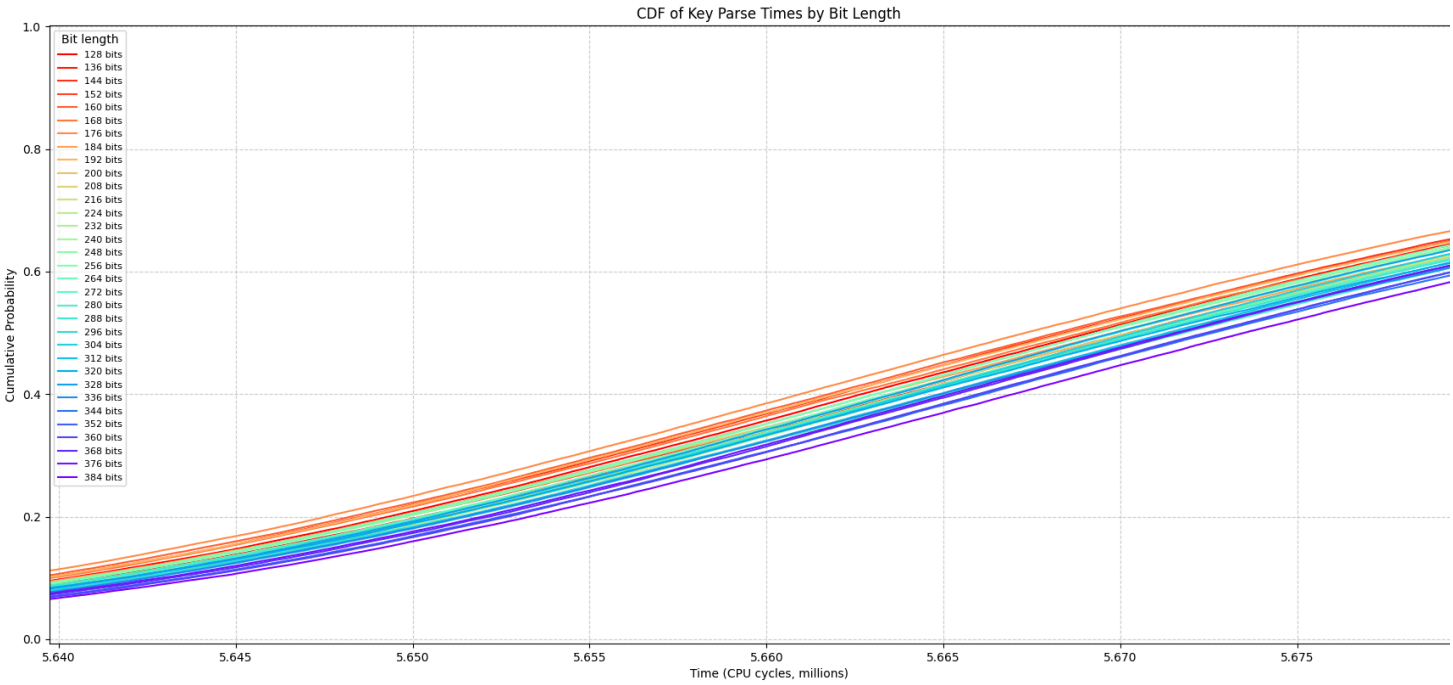
See how the distributions are separable (cf., Figure 1 Center in the paper):



visualize data: OpenSSL

```
python3 keydist.py keydist_openssl.dat
```

See how the distributions are not separable (lots of overlap, cf., Figure 1 Left in the paper):



terms

I'm releasing this research artifact to the community under the MIT License.

credit


```
Dr. Billy B. Brumley, D.Sc. (Tech.)
Director of Research, ESL Global Cybersecurity Institute (GCI)
Kevin O'Sullivan Endowed Professor, Department of Cybersecurity (CSEC)
Director, Platform Security Laboratory (PLATSEC)
Rochester Institute of Technology
Cybersecurity Hall 70-1770
100 Lomb Memorial Drive
Rochester, NY, 14623-5608, USA
S/MIME public key: https://people.rit.edu/bbbics/bbbics@rit.edu.crt
S/MIME public key: https://people.rit.edu/bbbics/bbb@iki.fi.crt
https://www.rit.edu/directory/bbbics-billy-brumley
https://www.rit.edu/cybersecurity/
```