

Let us walk on the 3-isogeny graph: Step-by-step Artifact Walkthrough

Jesús-Javier Chi-Domínguez, Eduardo Ochoa-Jiménez and
Ricardo-Neftalí Pontaza-Rodas

August 17, 2025

Outline

Repository Overview

System Requirements

How to Install and How to Build?

How to Run Tests?

How to Benchmark?

Reproducing Manuscript's Graphics

Generating Technical Documentation

CI/CD Pipeline Overview

How to download our Docker Container? (1/2)

Additional Resources: CPU benchmarking

License and Contributions

Repository Overview (1/4)

- ▶ *Let us walk on the 3-isogeny graph: efficient, fast, and simple* is an open-source C framework for using 3-radical isogenies to improve some post-quantum cryptosystems (dCTIDH + QFESTA).
- ▶ This presentation summarizes the software structure and reproducibility workflow.

Repository Overview (2/4)

- ▶ Hosted on GitHub: <https://github.com/Crypto-TII/pqc-engineering-ssec-23>
- ▶ Modular design with components: Presentation Video, System Requirements, Build, Test, Benchmarks, Docs, Manuscript results replication, and CI/CD Pipeline.

Repository Overview (3/4)

Overview of our paper - YouTube video:

<https://www.youtube.com/watch?v=BjedMooSV30&list=PLFgwYy6Y-xWYCFruq66CFXXiWEWckEk6Q>



Figure 1: Overview of our paper - YouTube video.

Repository Overview (4/4)

(Full) Guided Tour of our Artifact:

https://www.youtube.com/watch?v=hLk_B5NpKRA&list=PLFgwYy6Y-xWYCFruq66CFXXiWEWckEk6Q&index=10

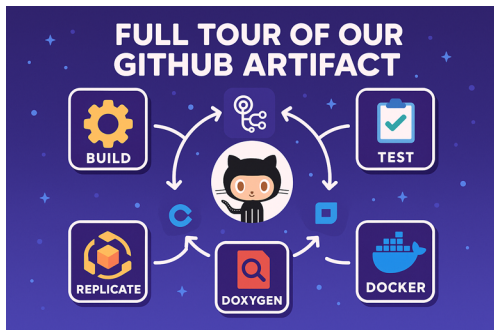


Figure 2: (Full) Guided Tour - YouTube video.

System Requirements

Our system requirements are extremely simple:

1. Out-of-the-box Linux (CPU Intel x86_64).
2. CMake + gcc
3. Python3:
 - ▶ Numpy
 - ▶ Matplotlib

How to Install?

Clone from GitHub.

Run:

```
git clone  
https://github.com/Crypto-TII/pqc-engineering-ssec-23.git
```

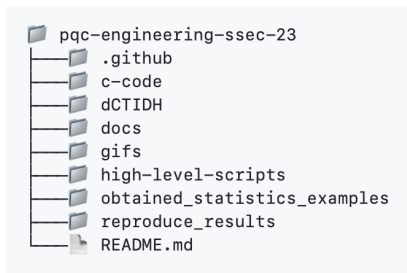


Figure 3: Downloaded project structure.

How to Build? (1/2)

Run:

```
cd c-code  
cmake -DCMAKE_BUILD_TYPE=Release -B cmake-build-release  
cd cmake-build-release  
make -j
```

Figure 4: Build instructions.

How to Build? (2/2)

```
liuend@liuend-aero16: ~/test
[ 76%] Building C object tests/CMakeFiles/tests-sssec-p592.dir/test_fq.c.o
[ 77%] Building C object benchmarks/CMakeFiles/benchmarks-sssec-p511.dir/benchmarks_main.c.o
[ 77%] Linking C executable benchmarks-sssec-p575
[ 77%] Linking C static library libsssec-p783.a
[ 78%] Building C object tests/CMakeFiles/tests-sssec-p511.dir/munit.c.o
[ 78%] Building C object tests/CMakeFiles/tests-sssec-p511.dir/test_main.c.o
[ 78%] Building C object tests/CMakeFiles/tests-sssec-p511.dir/test_fq.c.o
[ 79%] Building C object tests/CMakeFiles/tests-sssec-p511.dir/test_fp.c.o
[ 80%] Building C object tests/CMakeFiles/tests-sssec-p511.dir/test_isogeny_walks.c.o
[ 81%] Linking C executable benchmarks-sssec-p255
[ 82%] Linking C executable benchmarks-sssec-p383
[ 83%] Linking C executable benchmarks-sssec-p381
[ 84%] Linking C static library libsssec-p765.a
[ 84%] Built target ssec-p783
/usr/bin/ld: [ 84%] Building C object tests/CMakeFiles/tests-sssec-p783.dir/munit.c.o
warning: p254.s.o: missing .note.GNU-stack section implies executable stack
/usr/bin/ld: NOTE: This behaviour is deprecated and will be removed in a future version of the linker
[ 85%] Building C object tests/CMakeFiles/tests-sssec-p783.dir/test_fp.c.o
[ 85%] Linking C executable benchmarks-sssec-p398
[ 86%] Building C object tests/CMakeFiles/tests-sssec-p783.dir/test_main.c.o
[ 86%] Building C object tests/CMakeFiles/tests-sssec-p783.dir/test_fq.c.o
[ 87%] Building C object tests/CMakeFiles/tests-sssec-p783.dir/test_isogeny_walks.c.o
[ 87%] Building C object benchmarks/CMakeFiles/benchmarks-sssec-p783.dir/benchmarks_main.c.o
[ 87%] Built target ssec-p765
[ 88%] Building C object benchmarks/CMakeFiles/benchmarks-sssec-p765.dir/benchmarks_main.c.o
[ 88%] Built target benchmarks-sssec-p254
[ 89%] Building C object tests/CMakeFiles/tests-sssec-p765.dir/munit.c.o
[ 89%] Building C object tests/CMakeFiles/tests-sssec-p765.dir/test_main.c.o
[ 90%] Building C object tests/CMakeFiles/tests-sssec-p765.dir/test_fq.c.o
[ 90%] Building C object tests/CMakeFiles/tests-sssec-p765.dir/test_fp.c.o
[ 91%] Building C object tests/CMakeFiles/tests-sssec-p765.dir/test_isogeny_walks.c.o
/usr/bin/ld: warning: p575.s.o: missing .note.GNU-stack section implies executable stack
/usr/bin/ld: NOTE: This behaviour is deprecated and will be removed in a future version of the linker
[ 91%] Built target benchmarks-sssec-p575
```

Figure 5: Build process demo.

How to Run Tests? (1/2)

For running unit tests, simply execute:

- ▶ `cd cmake-build-release`
- ▶ `./tests/tests-ssec-p254`
- ▶ `./tests/tests-ssec-p255`
- ▶ `./tests/tests-ssec-p381`
- ▶ `./tests/tests-ssec-p383`
- ▶ `./tests/tests-ssec-p398`
- ▶ `./tests/tests-ssec-p511`
- ▶ `./tests/tests-ssec-p575`
- ▶ `./tests/tests-ssec-p592`
- ▶ `./tests/tests-ssec-p765`
- ▶ `./tests/tests-ssec-p783`

How to Run Tests? (2/2)

```
limuen@limuen-aero16: ~/test
tests/fp/cube_root      Total: [ 0.00992827 / 0.00981516 CPU ]
                        [ OK ] [ 0.00001464 / 0.00001443 CPU ]
tests/fq/is_zero        Total: [ 0.00732109 / 0.00721689 CPU ]
                        [ OK ] [ 0.00000041 / 0.00000040 CPU ]
tests/fq/locate_zero    Total: [ 0.00020398 / 0.00020008 CPU ]
                        [ OK ] [ 0.00034651 / 0.00034255 CPU ]
tests/fq/linear_pass    Total: [ 0.17325337 / 0.17127737 CPU ]
                        [ OK ] [ 0.00087555 / 0.00086566 CPU ]
tests/fq/add_and_sub    Total: [ 0.43777348 / 0.43282997 CPU ]
                        [ OK ] [ 0.00000081 / 0.00000080 CPU ]
tests/fq/mul_and_sqr    Total: [ 0.00040625 / 0.00040114 CPU ]
                        [ OK ] [ 0.00000165 / 0.00000163 CPU ]
tests/fq/inv            Total: [ 0.00082468 / 0.00081593 CPU ]
                        [ OK ] [ 0.00004197 / 0.00004155 CPU ]
tests/fq/batchinv       Total: [ 0.02098698 / 0.02077652 CPU ]
                        [ OK ] [ 0.00026474 / 0.00026214 CPU ]
tests/fq/square_root_slow Total: [ 0.13237001 / 0.13106988 CPU ]
                        [ OK ] [ 0.00034206 / 0.00033870 CPU ]
tests/fq/square_root_fast Total: [ 0.17103051 / 0.16934774 CPU ]
                        [ OK ] [ 0.00006664 / 0.00006599 CPU ]
tests/fq/cube_root      Total: [ 0.03331849 / 0.03299386 CPU ]
                        [ OK ] [ 0.00022046 / 0.00022002 CPU ]
tests/isogeny_walks/mul_by_small_constants Total: [ 0.11023042 / 0.11000877 CPU ]
                        [ OK ] [ 0.00000928 / 0.00000918 CPU ]
tests/isogeny_walks/degree_2 Total: [ 0.00463850 / 0.00459240 CPU ]
                        [ OK ] [ 0.02829769 / 0.02837272 CPU ]
tests/isogeny_walks/trit_string Total: [ 14.14884703 / 14.18636159 CPU ]
                        [ OK ] [ 0.00001137 / 0.00001158 CPU ]
tests/isogeny_walks/degree_3 Total: [ 0.00568370 / 0.00579185 CPU ]
                        [ OK ] [ 0.01205582 / 0.01215694 CPU ]
tests/isogeny_walks/degree_3_fp Total: [ 6.02790910 / 6.07846774 CPU ]
                        [ OK ] [ 0.24975477 / 0.24984901 CPU ]
                        Total: [ 124.87738506 / 124.92450647 CPU ]
22 of 22 (100%) tests successful, 0 (0%) test skipped.
limuen@limuen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-release$ |
```

Figure 6: Test demo.

How to Benchmark? (1/4)

- ▶ **Important:** Need to use flags `-DCMAKE_BUILD_TYPE=Release` `-DBENCHMARKING=CYCLES` when building.
- ▶ If the flags were not used, the benchmarks will be empty.

```
liuwen@liuwen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-release$ benchmarks/benchmarks-ssec-p254  
Numbers correspond for CGLHash2.  
Average: 0  
  
Q1: 0  
Median: 0  
Q3: 0  
  
Min: 0  
Max: 0  
  
Numbers correspond for CGLHash3.  
Average: 0  
  
Q1: 0  
Median: 0  
Q3: 0  
  
Min: 0  
Max: 0
```

Figure 7: Benchmark errors

How to Benchmark? (2/4)

To **build** benchmarking, simply execute:

- ▶ `cmake -DCMAKE_BUILD_TYPE=Release
-DBENCHMARKING=CYCLES -DARCHITECTURE=x8664 -B
cmake-build-release-cycles-x8664`
- ▶ `cd cmake-build-release-cycles-x8664`
- ▶ `make -j`

How to Benchmark? (3/4)

To **run** benchmarking, inside the `cmake-build-release-cycles-x8664` folder, simply execute:

- ▶ `benchmarks/benchmarks-ssec-p254`
- ▶ `benchmarks/benchmarks-ssec-p255`
- ▶ `benchmarks/benchmarks-ssec-p381`
- ▶ `benchmarks/benchmarks-ssec-p383`
- ▶ `benchmarks/benchmarks-ssec-p398`
- ▶ `benchmarks/benchmarks-ssec-p511`
- ▶ `benchmarks/benchmarks-ssec-p575`
- ▶ `benchmarks/benchmarks-ssec-p592`
- ▶ `benchmarks/benchmarks-ssec-p765`
- ▶ `benchmarks/benchmarks-ssec-p783`

How to Benchmark? (4/4)

```
liwuen@liwuen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-debug-cycles-x8664$ benchmarks/  
CMakeFiles/      benchmarks-ssec-p381  benchmarks-ssec-p511  benchmarks-ssec-p765  
benchmarks-ssec-p254  benchmarks-ssec-p383  benchmarks-ssec-p575  benchmarks-ssec-p783  
benchmarks-ssec-p255  benchmarks-ssec-p398  benchmarks-ssec-p592  
liwuen@liwuen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-debug-cycles-x8664$ benchmarks/benchmarks-ssec  
-p254  
  
Numbers correspond for CGLHash2.  
Average: 155026965  
  
Numbers correspond for CGLHash3.  
Average: 152084312  
  
liwuen@liwuen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-debug-cycles-x8664$ benchmarks/benchmarks-ssec  
-p255  
  
Numbers correspond for CGLHash2.  
Average: 155490448  
  
Numbers correspond for CGLHash3.  
Average: 134325940  
  
liwuen@liwuen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-debug-cycles-x8664$ benchmarks/benchmarks-ssec  
-p381  
  
Numbers correspond for CGLHash2.  
Average: 353393318  
  
Numbers correspond for CGLHash3.  
Average: 375604278  
  
liwuen@liwuen-aero16:~/test_demo/pqc-engineering-ssec-23/c-code/cmake-build-debug-cycles-x8664$
```

Figure 8: Benchmarking Demo

Reproducing Manuscript's Graphics

- ▶ Scripts located in `reproduce_results` folder.
- ▶ Need Python: Numpy and Matplotlib.

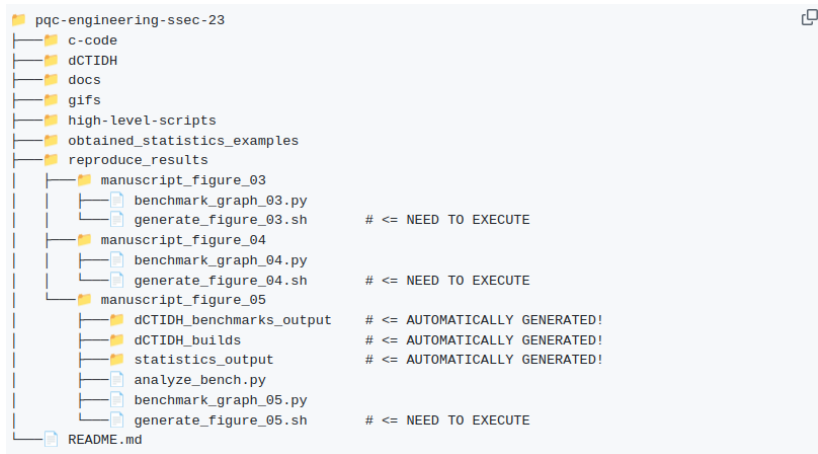


Figure 9: Location of bash scripts to reproduce manuscript's results.

Reproducing Manuscript's Graphics: Figure 3 (1/4)

Simply execute

- ▶ `cd reproduce_results/manuscript_figure_03`
- ▶ `chmod +x generate_figure_03.sh`
- ▶ `./generate_figure_03.sh`

Reproducing Manuscript's Graphics: Figure 3 (2/4)

```
ricardopontaza@ricardo-pontaza-PIGS-Linux: ~/demo/pqc-engineering-sscc-23/reproduce_results/manuscript_figure_03
Max: 745024488

Numbers correspond for CGLHash3.
Average: 769622965

Q1: 763647668
Median: 765412958
Q3: 768687464

Min: 760214504
Max: 955131572

benchmarks/benchmarks-sscc-p511 | tee benchmarks_sscc-p511-output.txt

Numbers correspond for CGLHash2.
Average: 1913451503

Q1: 1764669066
Median: 1771467675
Q3: 2188514140

Min: 1757381764
Max: 9349978846

Numbers correspond for CGLHash3.
Average: 2308021455

Q1: 2295673904
Median: 2304333139
Q3: 2316145304

Min: 2274135246
Max: 2589707834
```

Figure 10: Generation script for Figure 3.

Reproducing Manuscript's Graphics: Figure 3 (3/4)

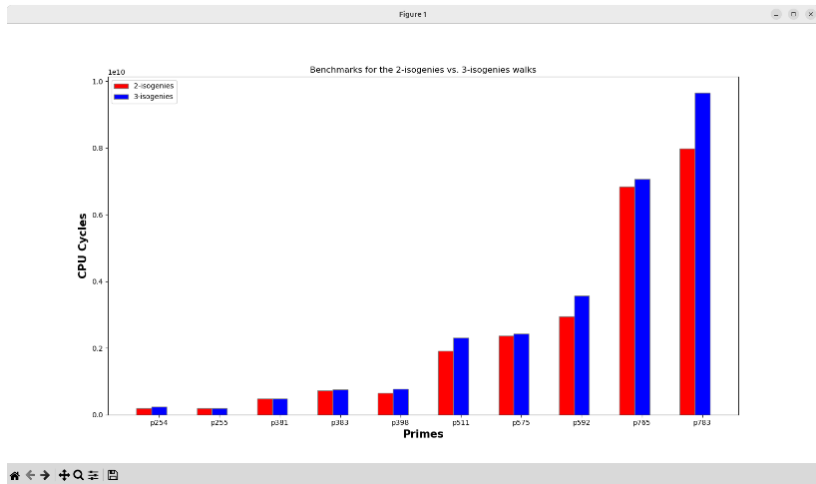


Figure 11: Generated statistical results from `generate_figure_03.sh`

Reproducing Manuscript's Graphics: Figure 3 (4/4)

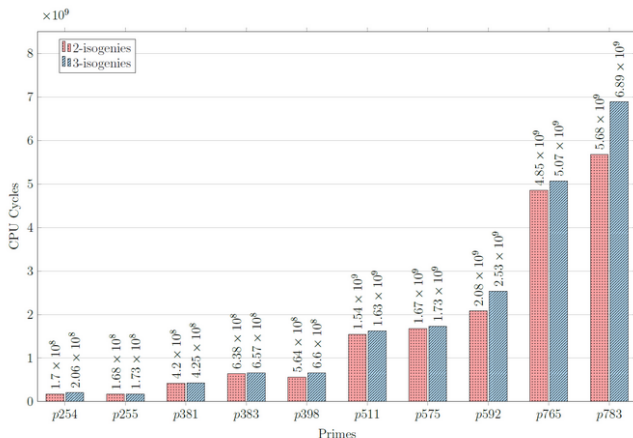


Figure 3: Benchmarks for the 2-isogenies vs. 3-isogenies walks, measured in CPU cycles.

Figure 12: Manuscript's Figure 3.

Reproducing Manuscript's Graphics: Figure 4 (1/4)

Simply execute

- ▶ `cd reproduce_results/manuscript_figure_04`
- ▶ `chmod +x generate_figure_04.sh`
- ▶ `./generate_figure_04.sh`

Reproducing Manuscript's Graphics: Figure 4 (2/4)

```
ricardopontaza@ricardo-pontaza-P1G5-Linux: ~/demo/pgc-engineering-ssec-23/reproduce_results/manuscript_figure_04
Numbers correspond for CGLHash2.
Average: 2368752706

Q1: 2358046436
Median: 2364409503
Q3: 2372597626

Min: 2339422190
Max: 2729886458

Numbers correspond for CGLHash3.
Average: 2443032114

Q1: 2431887688
Median: 2437604991
Q3: 2445513186

Min: 2415422662
Max: 2810899548

benchmarks/benchmarks-ssec-p592 | tee benchmarks_ssec-p592-output.txt

Numbers correspond for CGLHash2.
Average: 2940325885

Q1: 2928697408
Median: 2935591299
Q3: 2943246190

Min: 2904060464
Max: 3423052614

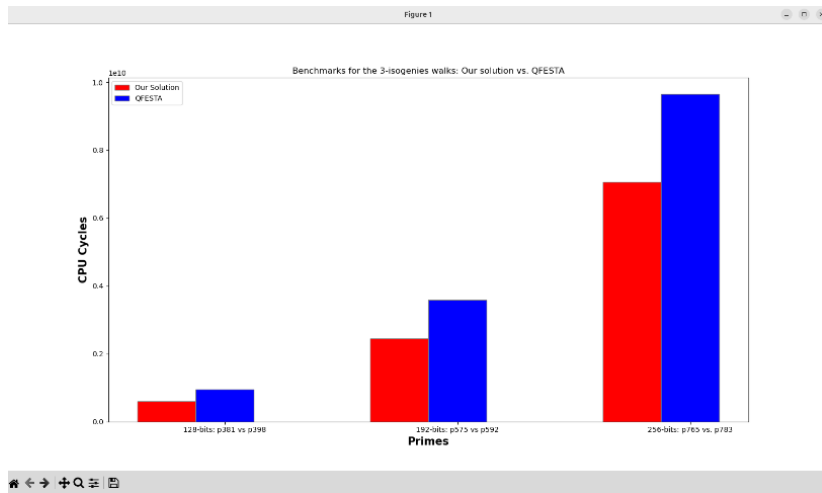
Numbers correspond for CGLHash3.
Average: 3583737646

Q1: 3564135148
Median: 3573225571
Q3: 3582969348

Min: 3538166126
Max: 4668548168
```

Figure 13: Generation script for Figure 4.

Reproducing Manuscript's Graphics: Figure 4 (3/4)



Reproducing Manuscript's Graphics: Figure 4 (4/4)

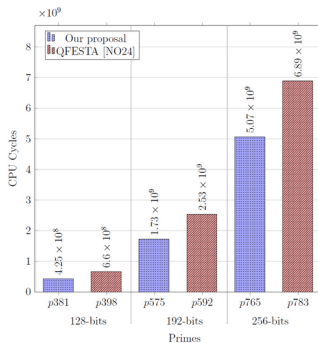


Figure 4: Benchmarks for the 3-isogenies walks for our proposed primes (p_{381} , p_{575} and p_{765}) vs. QFESTA [NO24] primes (p_{398} , p_{592} and p_{783}). Both p_{381} and p_{398} offer 128-bits security, while p_{575} and p_{592} offer 192-bits security, and p_{765} and p_{783} offer 256-bits security. For these six primes, the performance was measured in CPU cycles, having an improvement of 35,60% for 128-bits, 31,62% for 192-bits, and 26,41% for 256-bits, respectively.

Figure 15: Manuscript's Figure 4.

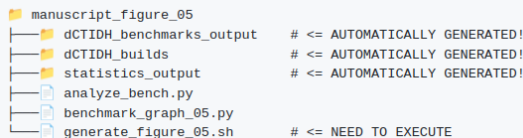
Reproducing Manuscript's Graphics: Figure 5 (1/5)

Simply execute

- ▶ `cd reproduce_results/manuscript_figure_05`
- ▶ `chmod +x generate_figure_05.sh`
- ▶ `./generate_figure_05.sh`

Reproducing Manuscript's Graphics: Figure 5 (2/5)

- ▶ The previous commands will (automatically) generate some folders.
- ▶ You can delete these (automatically-generated) folders between each run if necessary.



```
manuscript_figure_05
├── dCTIDH_benchmarks_output # <= AUTOMATICALLY GENERATED!
├── dCTIDH_builds            # <= AUTOMATICALLY GENERATED!
├── statistics_output        # <= AUTOMATICALLY GENERATED!
├── analyze_bench.py
├── benchmark_graph_05.py
└── generate_figure_05.sh    # <= NEED TO EXECUTE
```

Figure 16: Automatically-generated folders.

Reproducing Manuscript's Graphics: Figure 5 (3/5)

```
ricardopontaza@ricardo-pontaza-PIGS-Linux: ~/Desktop/projects_dev/pqc-engineering-sec-23/reproduce_results/manuscript_figure_05
ricardopontaza@ricardo-pontaza-PIGS-Linux: ~/Desktop/projects_dev/pqc-eng... ricardopontaza@ricardo-pontaza-PIGS-Linux: ~/Desktop/projects_dev/pqc-eng... ricardopontaza@ricardo-pontaza-PIGS-Linux: ~/Desktop/projects_dev/pqc-eng...
1 action experiments 100 Mcyc 194.741064+-0.99397 nulsq 331392+-0 sq 67846+-0 addsub 474404+-0 mul 768546+-0 conbo185 342543+-0
2 validate experiments 100 Mcyc 94.63172+-0.14898 nulsq 20623+-0 sq 8138+-0 addsub 17697+-0 mul 12493+-0 conbo185 19887+-0
3 action experiments 100 Mcyc 194.65050+-0.85961 nulsq 331392+-0 sq 67846+-0 addsub 474404+-0 mul 768546+-0 conbo185 342543+-0
4 validate experiments 100 Mcyc 94.63015+-0.13859 nulsq 20623+-0 sq 8138+-0 addsub 17697+-0 mul 12493+-0 conbo185 19887+-0
5 action experiments 100 Mcyc 194.52952+-0.95907 nulsq 331392+-0 sq 67846+-0 addsub 474404+-0 mul 768546+-0 conbo185 342543+-0
6 validate experiments 100 Mcyc 94.66897+-0.13845 nulsq 20673+-0 sq 8138+-0 addsub 17697+-0 mul 12493+-0 conbo185 19887+-0
7 action experiments 100 Mcyc 194.63947+-0.97411 nulsq 331392+-0 sq 67846+-0 addsub 474404+-0 mul 768546+-0 conbo185 342543+-0
8 validate experiments 100 Mcyc 94.65861+-0.13411 nulsq 20623+-0 sq 8138+-0 addsub 17697+-0 mul 12493+-0 conbo185 19887+-0
9 action experiments 100 Mcyc 194.59590+-0.87310 nulsq 331392+-0 sq 67846+-0 addsub 474404+-0 mul 768546+-0 conbo185 342543+-0
10 total validate experiments 1000 Mcyc 561.6561+-0.11650 nulsq 20623+-0 sq 8138+-0 addsub 17697+-0 mul 12493+-0 conbo185 19887+-0
11 total action experiments 1000 Mcyc 194.68513+-0.98189 nulsq 331392+-0 sq 67846+-0 addsub 474404+-0 mul 768546+-0 conbo185 342543+-0
12 echo -----
13
14 echo "Parsing benchmarking results for OPT_RAD 1 - 2047n61194 - keygen"
15 Parsing benchmarking results for OPT_RAD 1 - 2047n61194 - keygen
16 BENCH_KEYGEN_PEMF03-bench_keygen_194_opt_rad_1.out
17 BENCH_KEYGEN_PEMF03_PEMF03-analyzed_bench_keygen_194_opt_rad_1.txt
18 python3 analyze_bench.py
19 tee statistics_output/analyzed_bench_keygen_194_opt_rad_1.txt
20
21 validate experiments 100 Mcyc 86.10661+-0.09396 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
22 action experiments 100 Mcyc 10524.36349+-10811.98174 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
23 validate experiments 100 Mcyc 86.13284+-0.18497 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
24 action experiments 100 Mcyc 10621.91788+-10811.95350 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
25 validate experiments 100 Mcyc 86.12262+-0.11780 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
26 action experiments 100 Mcyc 10533.93626+-10816.46087 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
27 validate experiments 100 Mcyc 86.11994+-0.08849 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
28 action experiments 100 Mcyc 10517.91954+-10814.64857 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
29 validate experiments 100 Mcyc 86.11644+-0.08849 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
30 action experiments 100 Mcyc 10526.87631+-10813.51447 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
31 validate experiments 100 Mcyc 86.11708+-0.09771 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
32 action experiments 100 Mcyc 10526.77092+-10809.68648 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
33 validate experiments 100 Mcyc 86.12438+-2.48354 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
34 action experiments 100 Mcyc 10577.74266+-10819.80959 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
35 validate experiments 100 Mcyc 86.10117+-0.14065 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
36 action experiments 100 Mcyc 10529.39521+-10806.68094 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
37 validate experiments 100 Mcyc 86.10311+-0.11590 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
38 action experiments 100 Mcyc 10639.70787+-10811.44995 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
39 validate experiments 100 Mcyc 86.12801+-0.09072 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
40 action experiments 100 Mcyc 10538.38651+-10815.83557 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
41 total validate experiments 1000 Mcyc 86.11784+-0.78406 nulsq 19597+-0 sq 7805+-0 addsub 16133+-0 mul 11792+-0 conbo185 18843+-0
42 total action experiments 1000 Mcyc 10529.17175+-10813.37398 nulsq 2354323+-243011 sq 764178+-84148 addsub 2459914+-2396273 mul 1590154+-1588666 conbo185 2324485+-2381636
43 echo -----
44
45 echo "Parsing benchmarking results for OPT_RAD 1 - 2047n61194 - act"
46 Parsing benchmarking results for OPT_RAD 1 - 2047n61194 - act
47 BENCH_ACT_PEMF03-bench_act_194_opt_rad_1.out
48 BENCH_ACT_PEMF03_PEMF03-analyzed_bench_act_194_opt_rad_1.txt
49 python3 analyze_bench.py
50 tee statistics_output/analyzed_bench_act_194_opt_rad_1.txt
```

Figure 17: Generation script for Figure 5.

Reproducing Manuscript's Graphics: Figure 5 (4/5)

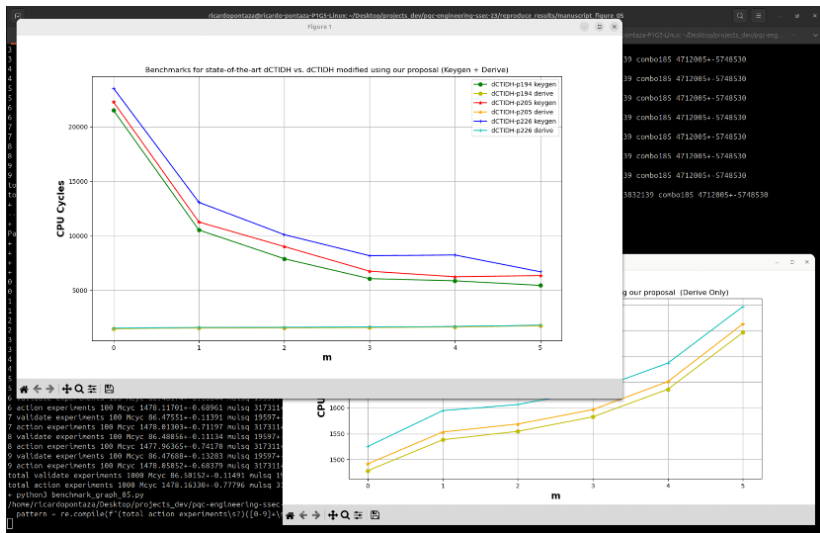


Figure 18: Generated statistical results from generate_figure_05.sh

Reproducing Manuscript's Graphics: Figure 5 (5/5)

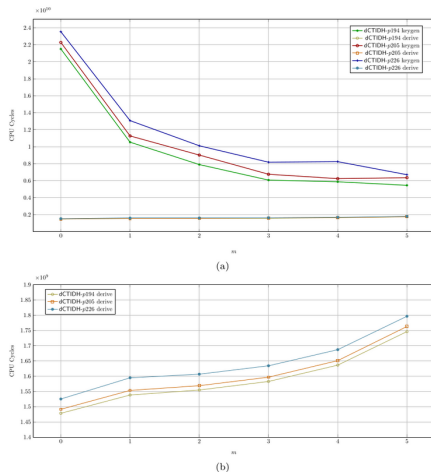


Figure 5: Benchmarks for state-of-the-art dCTIDH vs. dCTIDH modified using our proposal. Both the key generation (keygen) and the shared key derivation (derive) were tested. From

Figure 19: Manuscript's Figure 5.

Generating Technical Documentation (1/2)

We use Doxygen to generate the technical documentation.

- ▶ Configuration file: `Doxyfile`
- ▶ To generate, simply execute:
 - ▶ `cd docs`
 - ▶ `doxygen Doxyfile`
- ▶ Output in `docs/html/index.html`

Public link:

<https://crypto-tii.github.io/pqc-engineering-ssec-23/>

Generating Technical Documentation (2/2)

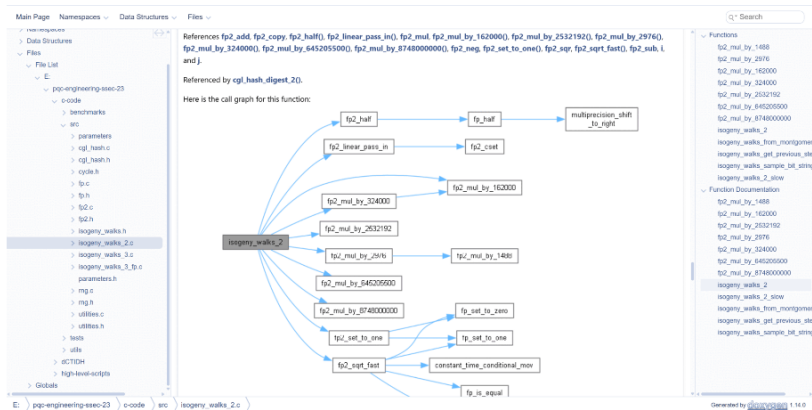


Figure 20: Technical documentation generated using Doxygen.

CI/CD Overview (1/3)

In order to show that our project can be integrated in a Real-World industrial environment, we provide a CI/CD pipeline.

- ▶ We use GitHub Actions for CI/CD.
- ▶ Pipeline includes **Build**, **Test**, **Benchmark**, and **Reporting** stages.
- ▶ YAML config:
`.github/workflows/cmake-multi-platform.yml`

CI/CD Overview (2/3)

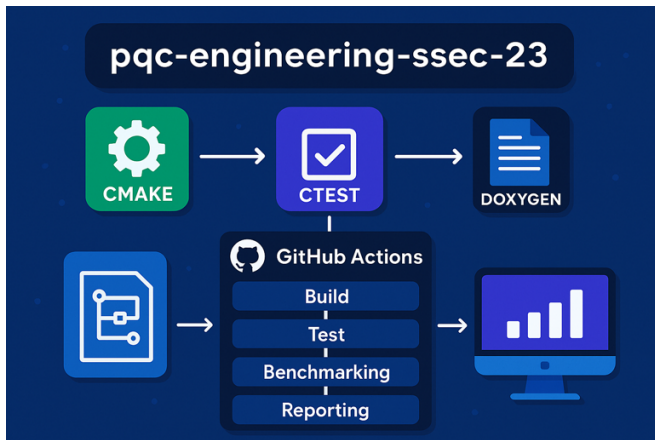


Figure 21: Designed CI/CD pipeline with **Build**, **Test**, **Benchmarking**, and **Reporting** stages.

CI/CD Overview (3/3)

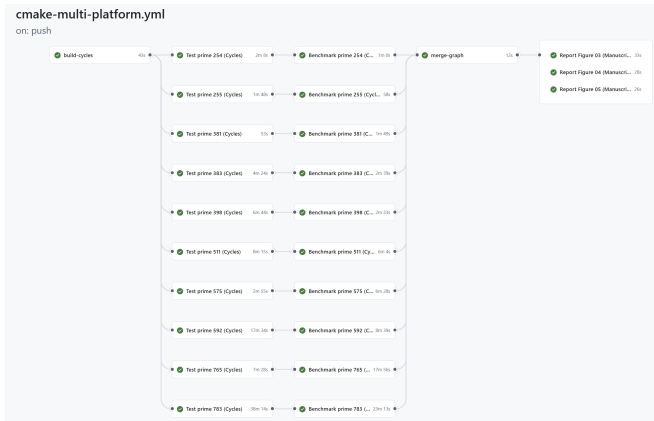


Figure 22: Pipeline in action, running with GitHub actions.

CI Stage: Build

- ▶ Triggers on push and pull request
- ▶ Any linux (Ubuntu example), Intel x86_64 CPU.
- ▶ Uses CMake caching for speed.

CI Stage: Test

- ▶ Runs unit and integration tests
- ▶ Stores artifacts for future analysis
- ▶ Automatic failure reports

CI Stage: Benchmark

- ▶ Executes performance benchmarks in both **CPU cycles** and **execution nanoseconds**.
- ▶ Benchmarking for every proposed prime.

CI Stage: Reporting (1/2)

- ▶ All the scripts used to reproduce our results reported in the manuscript are tested.
- ▶ The generated statistical data and the generated graphs are uploaded as public artifacts in our GitHub pipeline so they can be used freely.
- ▶ This allows collaborators, scientists, and anyone in general to reproduce, validate, and expand our research project.

CI Stage: Reporting (2/2)

Summary				
Job				
● build-cycles				
● Test prime 254 (Cycles)				
● Test prime 255 (Cycles)				
● Test prime 381 (Cycles)				
● Test prime 383 (Cycles)				
● Test prime 398 (Cycles)				
● Test prime 511 (Cycles)				
● Test prime 575 (Cycles)				
● Test prime 592 (Cycles)				
● Test prime 765 (Cycles)				
● Test prime 783 (Cycles)				
● Benchmark prime 254 (Cycles)				
● Benchmark prime 255 (Cycles)				
● Benchmark prime 381 (Cycles)				
● Benchmark prime 383 (Cycles)				
● Benchmark prime 398 (Cycles)				
● Benchmark prime 511 (Cycles)				
● Benchmark prime 575 (Cycles)				
● Benchmark prime 592 (Cycles)				
● Benchmark prime 765 (Cycles)				
● Benchmark prime 783 (Cycles)				
● merge-graph				
● Report figure 03 (Manuscript)				
● Report figure 04 (Manuscript)				
● Report figure 05 (Manuscript)				
Run details				
🔗 Usage				
📄 Workflow file				
Artifacts				
Produced during runtime				
Name	Size	Digest		
🔗 benchmarks_soc	9.23 KB	sha256:7173a555c5532afaa80ff7031944262c5cc0f564a83fa52f0440830c1...		📄 🔗
🔗 benchmarks_soc-p254-output	962 Bytes	sha256:fa51398ff8b75e4f60d64085123bce59f12bfda242f308f15a7318b732...		📄 🔗
🔗 benchmarks_soc-p255-output	952 Bytes	sha256:e73e486d4f031352f6f575fab3f6b243ca38f17a0f43c72a0d8171353...		📄 🔗
🔗 benchmarks_soc-p381-output	961 Bytes	sha256:ee587f8dc7045cc53247975ac6a387f4e7b2d0b5493c28c387d62bc0...		📄 🔗
🔗 benchmarks_soc-p383-output	962 Bytes	sha256:b2a5a302d32f7233f9220f86d86327c4dc0b3573a3c185c2804889b...		📄 🔗
🔗 benchmarks_soc-p398-output	968 Bytes	sha256:115087c380668226c7384277a25080f11807861506ac395b2684a65d...		📄 🔗
🔗 benchmarks_soc-p511-output	963 Bytes	sha256:03d46c737281703031433865324788bc75cc08072c5c17631d904...		📄 🔗
🔗 benchmarks_soc-p575-output	966 Bytes	sha256:e75c32f859a68338188c396080f280336a84b3386641c763c38752...		📄 🔗
🔗 benchmarks_soc-p592-output	970 Bytes	sha256:087c296d8e773259847f73ca08964a85684d70436d32c457732...		📄 🔗
🔗 benchmarks_soc-p765-output	971 Bytes	sha256:1a073095d345a56a7798061c329684d2865b542f37604976b703...		📄 🔗
🔗 benchmarks_soc-p783-output	973 Bytes	sha256:777177a388d272322c7091c84933131a2a3708a2ab6b18334833...		📄 🔗
🔗 cmake-build-release-cycles-0864	1.92 MB	sha256:da7545c1c1308081575d8d33bd408705105c3a3f215c493b089c...		📄 🔗
🔗 generated-figure-03	12.6 KB	sha256:03c78c5a159884bca08f06b768bd423f23c6f08808b37a2b2773d8...		📄 🔗
🔗 generated-figure-04	13.7 KB	sha256:a8c25e65f30ff71425d476a4d2d12328b803d549718b94732f4f55d...		📄 🔗
🔗 generated-figure-05	29.4 KB	sha256:c811c36128f780c18b2f25aaf07242d91e33476a4f13f8f8b9c379c55...		📄 🔗
🔗 reproduce-results-code	7.14 KB	sha256:19f2f0ff7f8f643bc5548962644876b27356a402548313a4822074b3...		📄 🔗

Figure 23: Publicly available artifacts.

Docker Container

Simply execute

- ▶ `docker pull tiicrc/github-selfhosted-runner-pqc:latest`
- ▶ `docker images | grep pqc`

Docker Container (2/2)

To mount, first locate your terminal at the artifact's root folder (*pqc-engineering-ssec-23*) and execute

- ▶ `docker run --rm -ti -v $PWD:/src -w /src tiicrc/github-selfhosted-runner-pqc:latest bash`

After mounting, the terminal will change to

- ▶ `/src# <insert commands here>`

Industrial Readiness Proof-of-Concept

- ▶ Simulates real deployment environments
- ▶ Documented logs, errors, and benchmarking outputs

Additional Resources: CPU benchmarking

- ▶ Included details on how to:
 - ▶ Turn off turbo-boost.
 - ▶ Assembly instructions used in our benchmarking.
- ▶ Automated benchmarking scripts under
`high-level-scripts/benchmark_02_20250408.sh`

License and Contributions

- ▶ Open-source under Apache License.
- ▶ License guidelines in LICENSE file.
- ▶ Issues and PRs welcome!

About the Authors

- ▶ Jesús-Javier Chi-Domínguez,
- ▶ Eduardo Ochoa-Jiménez,
- ▶ Ricardo-Neftalí Pontaza-Rodas.

Thank you

LET US WALK ON
THE 3-ISOGENY GRAPH:
EFFICIENT, FAST, AND SIMPLE



THANK YOU

Thank you

Thank you!