

8 Post-Quantum Cryptography (PQC) Security

Investigating PQC Hardware Security and Performance

A Introduction

With the advent of quantum computers, the current public-key cryptographic algorithms which are based on computationally hard problems **will be broken**. Public-key cryptographic algorithms are used for many purposes including key exchange/refreshment (for example based on the famous Diffie-Hellman key exchange algorithm), digital signature initiation/verification, and integrity/authentication benchmarks.

Among the important public-key cryptographic schemes are RSA which is based on integer factorization hardness of large composite numbers, and elliptic curve cryptography (ECC) which is based on the hardness of discrete logarithm problem for points on elliptic curves in binary or prime fields. *Although ECC is used as a replacement for RSA almost in all major applications nowadays because of its attractive implementations with the same level of security as RSA, it is prone to potential post-quantum attacks in the hands of attackers.*

Post-quantum cryptography (PQC) assumes that if and when we have powerful computers based on quantum principles, the hard problems that algorithms such as ECC and RSA are based on will be broken in polynomial time. This means all your credit card transactions, government top secret data, or private health data will be known to everyone. We cannot wait till such computers arrive and then reactively look for remedies!

National Institute of Standards and Technology (NIST) has had a world-wide competition to tackle this problem and as of 2022 has standardized a number of algorithms as substitutes to RSA and ECC which are post-quantum safe.

B Background

Symmetric key cryptography, e.g., Advanced Encryption Standard (AES) is not affected in post-quantum era. One can increase the key size to 256 bits and although the famous Grover's quantum algorithm (for symmetric key schemes) reduces its security, we will be still good. The problem of post-quantum security is the algorithms used for public-key cryptography (RSA and ECC) which will be broken based on the famous Shor's quantum algorithm in polynomial time.

The NIST standardization has been investigating possible replacement for quantum computer attack-prone ECC and RSA for many years. The winners (called standards) have been listed in the Table 1 where **key encapsulation mechanisms (KEMs)** and **public-key encryption (PKE)** algorithms as well as those used for providing **digital signatures** have been tabulated. The KEMs are used for exchanging the keys that are used in the symmetric key cryptography, whereas digital signatures first sign a document/message and then anyone can verify the authenticity of the message and its integrity through publicly known parameters and a set of keys. The listed algorithm Kyber (for KEM) is based on structured lattices. The signatures are also based on hard problems which are resistant to quantum attacks (Dilithium, Falcon, SPHINCS+).

PKE/KEM	Digital Signatures
CRYSTALS-Kyber	CRYSTALS-Dilithium
	Falcon
	SPHINCS+

Table 1: NIST PQC Standards

Reading Check


1. Why do we need PQC? Why do we need to start looking for PQC now rather than later?
2. What's the difference between Key Encapsulation Mechanism (KEM) and Digital Signatures? (we use these terms in both classic and PQC)
3. What is difference between Grover's and Shor's quantum algorithms?

C Hardware and Software Tools

We will be using AMD Xilinx Vivado v2021.1-2022.2 to compare a classic and a PQC algorithm in terms of performance and implementation metrics. You will be provided with the RTL of ECC and the PQC algorithm Kyber as a case study. The reason to assess this is to show that the PQC algorithms are also practical in embedded architectures, noting that the hardware and software implementations are improving. We will also use a web link for testing the Shor's quantum algorithm. No other tools are required.

D Getting Started

We need the RTL/HDL of ECC for this lab to be used in Part B. Thus, download the PHS-Lab-08 directory which contains the files you need to implement.

PHS-Lab-08
 ECC-HDL

E Lab Assignment

This lab has two distinct parts. First you become familiar with Shor's quantum algorithm to break the current public key cryptographic algorithms. This serves as motivation for the second part which is to compare a classic cryptographic algorithm (ECC) and PQC hardware implementations, implemented on a Xilinx FPGA family/device.

Part A: Shor's Quantum Algorithm

Shor's algorithm is a polynomial-time quantum computer algorithm for integer factorization. Informally, it solves the following problem: Given an integer " n ", find its prime factors. It was invented in 1994 by the American mathematician Peter Shor. Through this quantum algorithm, the current public-key cryptography algorithms such as RSA and ECC will be broken when we have quantum computers. Examples of successful factoring have been 15 (3 times 5) in 2001 and 21 (3

times 7) in 2012. In 2019, an attempt was made to factor the number 35 (5 times 7) using Shor's algorithm on an IBM Q System One, but the algorithm failed because of accumulating errors. As you may know though, in reality, we deal with much larger numbers in public-key cryptography (more than 3000-bit numbers for RSA and a couple of hundred bits for ECC). Thus, we need much more advanced quantum computers; nevertheless, such powerful computers may be invented in few years.

- 1) Read and understand both the quantum part and the classic part of Shor's algorithm from a source, for example, https://en.wikipedia.org/wiki/Shor's_algorithm.

In this part of the lab, we would like to use a simple example to break public-key cryptography (PKC) using Shor's quantum algorithm. Let us use a simulator to process this.

- 2) Go to <https://blendmaster.github.io/ShorJS/>
- 3) Choose an odd number to factor for " n ", for example let us use the number 15 (as you know an even composite number of two factors is easy to factor). Due to the limitations of the simulation, " n " must be less than 30.
- 4) Classic Part
 - a) Choose a random number " a " or let the algorithm choose one, this has to be less than " n ".
 - b) Try different values of " a ", and understand the classic part and note that in reality we deal with much larger numbers so we do need the quantum part.
 - c) Once you find a co-prime " n " and " a " pair for the classical part, then move on to the quantum part. " n " and " a " are co-prime if $GCD(n, a) = 1$.
- 5) Quantum Part
 - a) For our small example of 15, we need 8-qubits. Investigate the initial and transform quantum registers.
 - b) Investigate the graphs and also the final Fourier Transform register.
 - c) Continue with post-processing to get to the factors.
- 6) Classical Post Processing
 - a) Read and understand how the post processing part works.
 - b) Make sure you get the correct factors. If not, select "retry" or "retry quantum part" to try again the quantum part until you get a valid answer. Alternatively, try a different value for " a ".
 - c) Once you get the correct factors, take a screenshot of the output.
- 7) Use different numbers " n " and " a " in the simulator. Report the results and explain why the scheme might be successful and/or unsuccessful.

Part B: Classic vs. PQC Algorithms on Hardware

In this part, we compare the FPGA areas of the most prominent classical public-key cryptography based on ECC with those of PQC finalists. ECC is being used in the most secure applications nowadays because of its efficient implementations on hardware and software platforms, e.g., ARM/RISC-V processors and FPGA/ASIC. Nevertheless, it is going to be vulnerable after quantum computers are developed for attacking the hard problems it is based on.

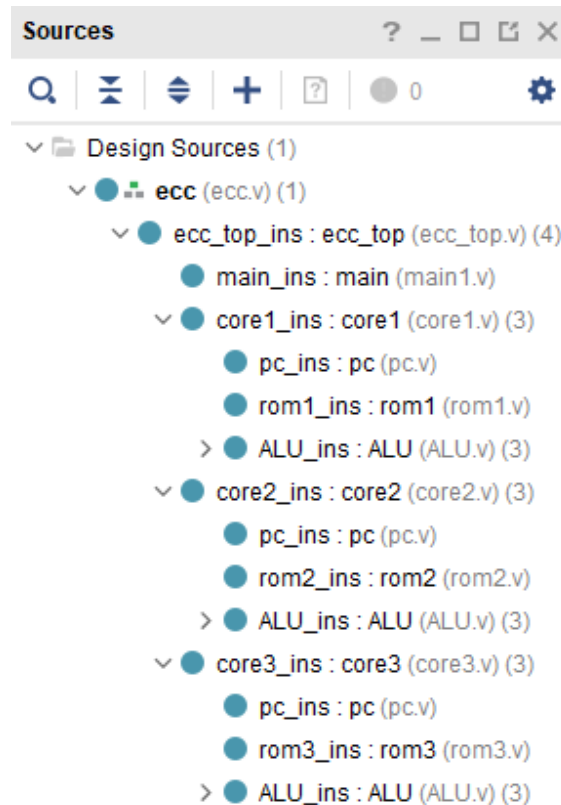


Figure 1: A quick view of tree-like RTL

- 1) Read and understand the basics of ECC and its elliptic-curve-based discrete logarithm problems form a source, for example, https://en.wikipedia.org/wiki/Elliptic-curve_cryptography.
- 2) Implement (you need to synthesize and then implement in the tool) the provided RTL (Verilog) top-level entity and all other entities in AMD Xilinx Vivado v2021.1-2022.2 tool choosing Artix-7 FPGA family. This ECC is over finite field $GF(2^{163})$ with HDL implementation by using three finite field cores. Make sure you use FPGA part "XC7A200TFFG1156-1". If all source files are read by the tool under your new project, the tree-like RTL would look like Figure 1.
- 3) Under Map Report in AMD Xilinx Vivado v2021.1-2022.2, what are the total number of "Slices", "LUTs" (add the "LUTs as Logic" and "LUTs as Memory"), and "Slice Registers"?
- 4) Compare this with those of Kyber512 as reported by the authors in 2021:

<https://tches.iacr.org/index.php/TCHES/article/view/8797/8397>

Note: Look for "LUT", "FF", and "Slice" in the publication above which correspond to "LUTs" ("LUTs as Logic" + "LUTs as Memory"), "Slice Registers", and "Slices" in AMD Xilinx Vivado v2021.1-2022.2, respectively.

What do you conclude? Note that ECC will be vulnerable to quantum attacks but these standards will not, so that would be a prominent advantage for the NIST winner Kyber.

F Badges and Achievements

This experiment involves **cryptography**, **Chipwhisperer**, and **data visualization**. By successfully completing the lab you can unlock the following achievements:

- CRYPTO3: Cryptography
- C4: Chipwhisperer
- STAT6: Statistical Analysis

Depending on the demonstrated skill, each achievement will be further divided into bronze, silver, and gold levels.

G Deliverables

Follow the instructions in Appendix A for writing the lab report. You need to add all your findings in both parts. The report should also contain the following:

- Answers to the reading check questions.
- Briefly describe the "Classical Part" from Part A. How does it work?
- Briefly explain the purpose of the "Quantum Part" from Part A.
- Screenshot of the "Classical Post Processing" output from part A.
- Comparison table for Part B.