Towards Cloud-based Infrastructure for Post-Quantum Cryptography Side-channel Attack Analysis

Tristen Teague
Electrical Engineering and Computer Science
University of Arkansas
Fayetteville, Arkansas, USA
tdteague@uark.edu

Mayeesha Mahzabin
Electrical Engineering and Computer Science
University of Arkansas
Fayetteville, Arkansas, USA
mahzabin@uark.edu

Alexander Nelson
Electrical Engineering and Computer Science
University of Arkansas
Fayetteville, Arkansas, USA
ahnelson@uark.edu

David Andrews
Electrical Engineering and Computer Science
University of Arkansas
Fayetteville, Arkansas, USA
dandrews@uark.edu

Miaoqing Huang
Electrical Engineering and Computer Science
University of Arkansas
Fayetteville, Arkansas, USA
mqhuang@uark.edu

***Abstract*—Post-quantum cryptography (PQC) refers to cryptographic algorithms that are thought to be secure against a cryptanalytic attack by a quantum computer. Before PQC algorithms can be widely deployed to replace the current standards such as the RSA algorithm, they need to be rigorously evaluated theoretically and practically. In this work, we present a cloud-based infrastructure being developed for performing side-channel analysis on PQC algorithms for the research community. Multiple types of side-channel attacks, such as timing attacks, power attacks, and electromagnetic attacks can be applied on different types of devices, such as FPGA devices and microcontrollers. An automated tool flow is being developed that can run executables on the target devices, collect traces (e.g., power consumption waveforms and electromagnetic radiation signals), perform leakage assessment (using Test Vector Leakage Assessment), and generate analysis reports. Remote users access the infrastructure through a web portal by uploading the hardware or software implementations of cryptographic algorithms. Side-channel attack and leakage analysis are performed on the given implementation. Finally, the user is informed for downloading the analysis report from the portal.**

***Index Terms*—Side-channel attack, Post-Quantum cryptography, Leakage detection, Hardware design, Cloud-based Infrastructure.**

I. Introduction

The modern internet enables secure multi-party communication through cryptography. Asymmetric cryptography allows these communications to be established without prior arranged keys. However, current standardized asymmetric encryption schemes are subject to key factorization attacks in polynomial time when using a quantum computer. Standardization efforts are underway to create quantum-safe asymmetric cryptography. The algorithms and their implementations need to be thoroughly vetted for side-channel resistance. *The overall objective of this project is to create an open research infrastructure that helps improve the side-channel analysis of the implementations of Post-Quantum Cryptography that will be used for secure communication among the digital infrastructure.*

Quantum Computers and Cryptography: One consequence of Shor's algorithm [1] is that discrete logarithm and factoring can be performed in many circumstances in polynomial time. This renders most standardized asymmetric cryptographic algorithms (such as RSA or ECC) with significantly reduced security. To remedy this oncoming concern, the National Institute of Standards and Technology (NIST) [2] began a standardization effort in 2016 to identify and decide on asymmetric algorithms and their respective parameter sets that are resistant against both classical and quantum computers. This effort is named "Post Quantum Cryptography" or PQC. CRYSTALS-Kyber [3] has been selected and is undergoing final preparation for standardization as the key-establishment mechanism (KEM) of choice. SImilarly, for digital signatures, CRYSTALS-Dilithium [4], FALCON [5], and SPHINCS+ [6] have been selected for further standardization.

Cryptography and Side Channels: However, a physical implementation of these algorithms that are currently mathematically secure may, under certain circumstances, be physically insecure. The implementation of the algorithm (such AES on an embedded system) can leak secret information, including potentially its private key. Attacks that exploit the implementation rather than the design of the algorithm are called "side-channel attacks" (SCA). Some examples of SCA involve measuring power, electromagnetic radiation, and timing differences. All of these side-channels can yield additional information about the algorithm that should be secret. Therefore, it is crucial that the algorithm is mathematically secure and also SCA secure to ensure that secure communication is preserved.

PQC algorithms that are selected for standardization are either new or under-researched in SCA. Given that there are

978-1-5386-5541-2/22/$31.00 ©2022 IEEE

no established implementations, the existing implementations of PQC have been shown to have vulnerabilities [7] [8] [9] [10]. Performing side-channel analysis on cryptography is non-trivial. It requires specific domain knowledge, equipment, and resources. It can also be prohibitively expensive. Therefore, the goal of this project is to significantly reduce the barrier-to-entry to enable SCA research for PQC more broadly. We hope that by making this available, the broader security community will be able to more expediently arrive at secure and trusted PQC implementations.

Existing SCA Labs: There exist industry SCA labs that do conduct side-channel analysis. Some examples of SCA labs are Riscure with Inspector, Secure IC with Analyzr, Rambus offering DPA workstation, and eShard. Some of the companies provide software tools to conduct SCA, some provide auditing on the customer's implementation, and some also provide full workstations that conduct SCA. Unfortunately, the industrial solutions can be costly. Further there exists no SCA infrastructure open to the public for PQC algorithms. Riscure and PQShield are working towards an SCA lab dedicated to PQC algorithms. But it is unclear if it will be open to all or just customers who want SCA auditing. There is work of an open-source SCA platform named FOBOS2 [11]. However, it is geared towards lightweight ciphers and hardware implementations.

Regarding Power Electronics: Microgrid and distributed PV delivery has potentially pushed vital infrastructure into physically insecure locations. Therefore, it is vital that the community integrates safe implementations of PQC algorithms into the power infrastructure. Without secure PQC implementations, secure communication will not be guaranteed. Malicious actors can find and act on vulnerabilities on the PQC implementations, possibly disrupting the power grid or causing harm. Therefore, the plan of this work is to use this infrastructure to help test and validate the SCA security of new PQC implementations that will be integrated into the power electronics infrastructure.

In this work, we present the prototype of a multi-target and multi-tool power side-channel analysis platform that conducts the Test Vector Leakage Assessment (TVLA) on PQC implementations. This infrastructure offers SCA leakage assessments on PQC implementations. It is open to both industry and researchers who want to test the SCA security without requiring the necessary equipment. The current infrastructure contains two types of power trace collecting devices, i.e., high-end oscilloscopes for high-resolution trace collection, and NewAE ChipWhisperer for faster and smaller trace captures. Hardware and software implementations of PQC algorithms run on FPGA devices and various target boards (e.g., microcontroller boards), respectively.

II. The Overall Architecture

A. Main Components

The overview of the architecture for the prototype of the infrastructure is shown in Fig. 1. It consists of four main components. The client interface is responsible for servicing

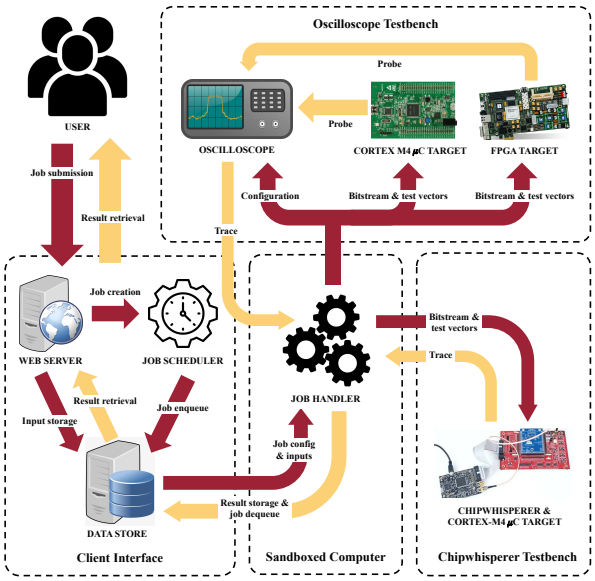


Fig. 1: An overview of the major system components.

the user, scheduling SCA jobs (similar to an HPC system), and storing the SCA analysis results from the lab computer. The sandbox computer consists of a lab computer connecting to either the ChipWhisperer testbench or the oscilloscope testbench. This computer will have a set of scripts (both in bash scripts and Python) that will control the target boards, control the oscilloscope or ChipWhisperer, and also process the power traces and compute statistical analysis. Once the job is completed, it will return the results back to the client interface.

The ChipWhisperer testbench consists of a ChipWhisperer-Lite connected to a CW308 UFO board. The ChipWhisperer is a hardware security tool performing side-channel analysis through power analysis and fault injections on either hardware or software victim boards. In the current prototype, the ChipWhisperer tool is utilized for software only targets for measurement. For the software targets, the STM32F4 microcontroller board is employed, which consists of an ARM Cortex-M4 processor (at 7.37 MHz). The victim board is controlled by the lab computer in the sandbox computer component. The ChipWhisperer testbench uses synchronous sampling tied to the victim board, requiring a lower sampling rate to be equivalent to a high asynchronous sampling oscilloscope [12]. The main drawback of ChipWhisperer-Lite is the relatively small sample buffer (24,400 samples total). Therefore, the prototype will only utilize the ChipWhisperer testbench for software implementations that can fit the whole measurement of the point of interest into the sample buffer. The inputs (public key, private key, plaintext) are all generated on the board.

The oscilloscope testbench consists of a Tektronix MDO34 3-BW-200 oscilloscope, a victim microcontroller for the software implementations, and a Xilinx Virtex-7 FPGA VC707

(at 100 MHz) for the hardware implementation. For software implementations, the shunt resistor found on the CW308 UFO board can be measured with a passive probe at various sampling rates and lengths depending on the length of the implementation runtime. For hardware implementations, a passive probe is attached to the output of VCCINT power rail. The victim boards in this testbench are controlled by the lab computer found in the sandbox computer component. The inputs, such as the ciphertext and the keys, are genereated on the FPGA's MicroBlaze processor. The sandbox computer utilizes a custom communication protocol [13] through Python to generate the inputs and feed them into the encryption/decryption module, which runs on the FPGA device.

B. Evaluation

Test Vector leakage Assessment (TVLA) [14] has become the de facto standard in the evaluation of side-channel measurements, such as power and electromagnetic (EM) traces. TVLA identifies differences between two sets of side-channel measurements by computing the uni-variate Welch's t-test for the two sets of measurements. The test can be employed to detect side-channel leakages that are not associated with any specific leakage model [15]. Two sets of measurements are taken, the first with fixed inputs and the second with random inputs, which will be referred to as T_f and T_r, respectively. The fixed set contains fixed variables utilized for the cryptographic operation. The random dataset will share all the same variables with the fixed set except for a single random variable. At each time step a pass/fail decision is given by testing for a null hypothesis such that the means of the two sets are equivalent. The TVLA at each time step is calculated as follows:

$$TVLA = \frac{\mu_r - \mu_f}{\sqrt{\frac{\sigma_r^2}{n_r} + \frac{\sigma_f^2}{n_f}}}, \quad (1)$$

where μ_r, σ_r, and n_r are the mean, standard deviation, and number of traces collected for T_r, respectively; likewise for trace set T_f. The null hypothesis is rejected with a confidence level of 99.9999% if the absolute value of the t-test is greater than 4.5 [14]. Assuming independent leakage, any variation in the power traces is the result of the underlying computation other than factors such as hardware architecture features [16]. Therefore, a rejected null hypothesis, which constitutes a fail decision, suggests that the two trace sets T_r and T_f are different and as such might leak some information about the underlying computation with the differing variables. An accepted null hypothesis suggests with high confidence that there is no effect on the power with the difference of the inputs and the use of variables.

Distinguishers are classes that test for the difference in power when varying certain variables associated with the distinguisher. Since there are many variables within PQC algorithms, the prototype must create many distinguishers to test each effect of the variables on the power traces. Within this work, the ciphertext distinguisher is utilized for the key encapsulation mechanism (KEM) algorithm and the message

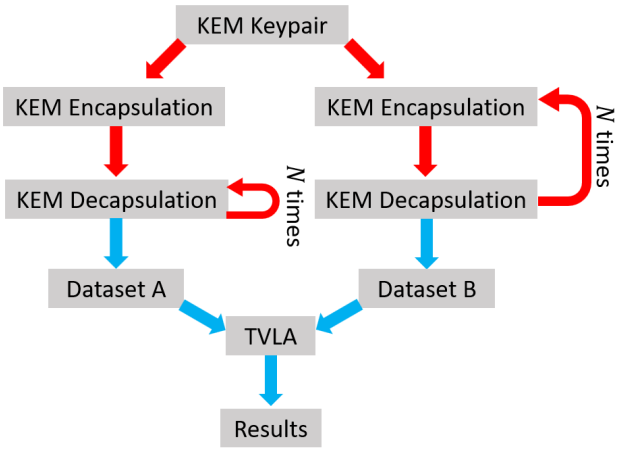


Fig. 2: TVLA Process for KEM Distinguisher.

distinguisher for the Public-Key Encryption (PKE) algorithm. The ciphertext distinguisher consists of two datasets, where one setup uses the same generated ciphertext (fixed), and the other generates a new ciphertext (random) every measurement. This distinguisher measures the power difference within the decapsulation sub-algorithm to see if varying the input (which is the ciphertext) affects the power. The message distinguisher will test to see if there is a power difference with the use of the message on the decryption sub-algorithm. The message distinguisher will contain two sets of data, where the fixed dataset has the same message used for all measurements. The random dataset will consist of a random message for every measurement.

The test vectors for the two test benches for the TVLA process are generated differently. For the software side implementations, the test vectors are generated in KEM Keypair and KEM Encapsulation. Dataset A consists of power traces utilizing the same generated ciphertext. Dataset B consists of power traces that have a randomly generated ciphertext. Both dataset A and B are collected during the same process, where the prototype first captures dataset A and then dataset B in the same job process. KEM keypair is ran first to generate the public and private key pair. This key will stay constant for both dataset A and B. For dataset A, encapsulation is ran once, and it will generate the fixed ciphertext that will be utilized as many times as the TVLA process requires in decapsulation. The point-of-interest is placed by the user in any place the user wants to measure inside decapsulation. For dataset B, encapsulation is ran every measurement to generate a new ciphertext, where this random ciphertext that was generated will be utilized only once in a decapsulation call. Once the testbench is finished measuring, both datasets are sent to the sandbox computer where it will calculate the TVLA results. The software side TVLA KEM distinguisher process can be seen in Fig. 2. For the hardware side, the inputs, such as the private key and the ciphertexts, are fed into the FPGA where it will perform decryption. It will follow a similar setup where dataset A has a fixed ciphertext and dataset B will

have a random ciphertext with many decryption runs where the measurement occurs. The keypair and the ciphertext test vectors are generated using Kyber's source C code.

```c
//Follows PQClean library
void custom_crypto_kem_keypair(unsigned char *pk,
    unsigned char *sk)
{
    //User's keypair following API
    crypto_kem_keypair(pk, sk);
}
```

(a) Unmasked Interface.

```c
static unsigned char pk_custom[CRYPTO_PUBLICKEYBYTES];
static masked_sk masked_sk_a;
void custom_crypto_kem_keypair() {
    //User's keypair function with custom variables
    crypto_kem_keypair(pk_custom, &masked_sk_a);
}
```

(b) Masked Interface.

Fig. 3: Software PQC Interfaces.

C. Taking User's Implementation

To take arbitrary PQC KEM implementations for the software side, the user must implement the sub-algorithms of the KEM process (keypair, encapsulation, and decapsulation). Once the sub-algorithms are implemented, they will be called by using the PQClean [17] API interface for simplicity. The unmasked interface can be seen with Fig. 3a. Since there exists implementations with countermeasures (such as masking) that utilize custom variables that do not follow the PQClean API interface, a separate interface is made for the user if specified. This masked interface can be seen in Fig. 3b. For the hardware side, the prototype will obtain the user's bitstream and testvectors through the client interface and will be utilized on the FPGA board.

D. Measuring Point of Interest

Previous asymmetric cryptography algorithms (such as RSA) were less complex than the newly developed standardized PQC algorithms. Many of the newly introduced standardized PQC algorithms contain many sub-algorithms, which make the run-time on platforms longer. Therefore, it may be difficult to measure the full algorithm in a single measurement since it will require a very large sample buffer. Therefore, it is up to the user to decide the area of measurement that will be performed for the SCA analysis. An example of the function measured in decapsulation in the prototype and how it placed the point of interest can be seen in Fig. 4.

III. Implementation and Results

With this prototype, a set of software tools have been developed to automate the power-based side-channel attack and analysis, as shown in Fig. 5. Starting from the hardware or software implementation of the algorithm uploaded by the user, a sequence of operations will take place in order, represented by numbers 1 to 5 in Fig. 5. The following is an explanation of each step.

```c
trigger_high();
poly_sub(&mp, v, &mp);
trigger_low();
```

Fig. 4: Area of Measurement.

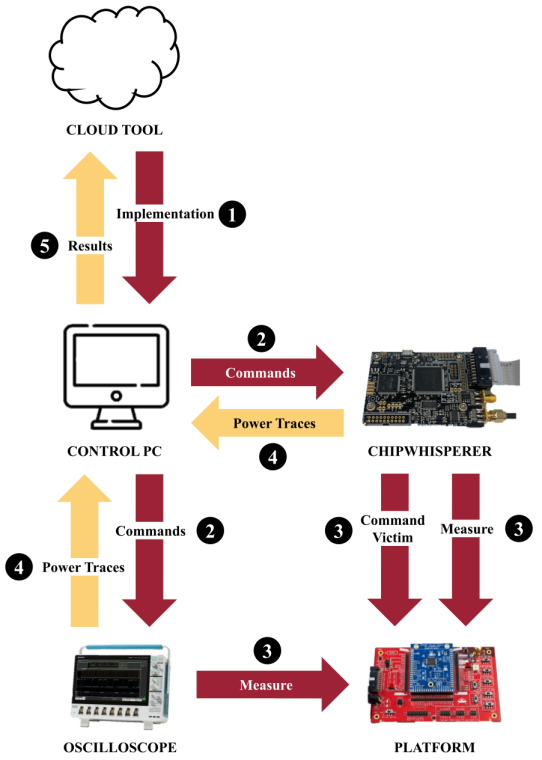
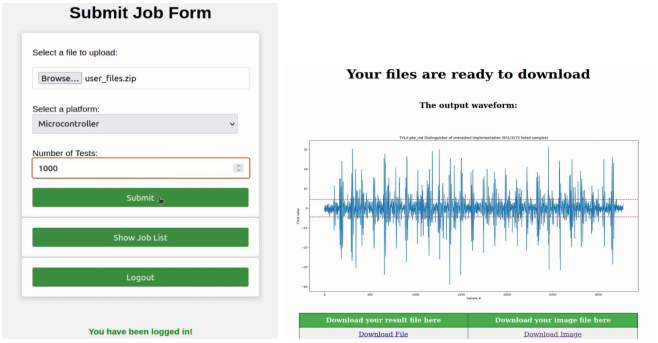


Fig. 5: An automated process for performing power-based side-channel attack and analysis.



(a) Job form on the client interface.

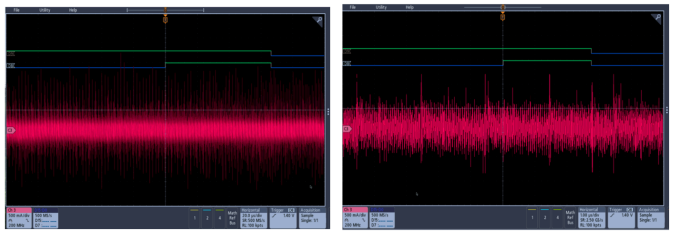
(b) TVLA result of an unmasked Kyber implementation.

Fig. 6: Client Interface.

1) The user sends the PQC implementation in either a bitstream for the hardware implementation or the source code that implements the sub-algorithms for the software implementation. The user will specify what platform they require to run the tests on, and the number of traces wanted. This can be seen in Fig. 6a. The web portal will then send the implementation to the appropriate lab computer to perform analysis on the testbench.
2) The control PC (i.e., the lab computer) uses Python script to control either the oscilloscope or the ChipWhisperer. The oscilloscope is always used for the hardware implementation. It is optionally employed for

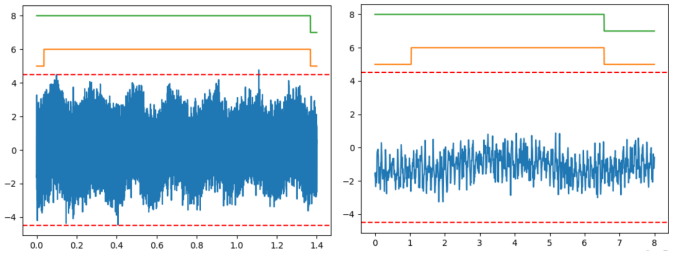
the software implementation if the algorithm requires high resolution traces or the sample buffer of the ChipWhisperer cannot save all the data.

3) For the case of the software implementation, the ChipWhisperer capture board (which is commanded by the control PC) is utilized to command the victim to perform cryptographic sub-algorithms followed by the TVLA process. In the hardware implementation case, the control PC sends inputs to the platform, and the oscilloscope measures the power consumption of the operations.
4) For each power measurement, the resulting power trace back is sent to the control PC. Once all the power measurements are done, the control PC performs the TVLA on the fixed dataset and the random dataset. It will then take these results from the TVLA and convert them into both a readable form and a graph.
5) The TVLA results will be sent back to the user on the client interface. One example can be seen in Fig. 6b.



(a) Power trace of the unmasked Kyber.

(b) Power trace of the masked Kyber.



(c) TVLA result of the unmasked Kyber.

(d) TVLA result of the masked Kyber.

Fig. 7: Side-channel attack and analysis of CRYSTALS-Kyber on Xilinx Virtex-7 FPGA.

As a showcase, we implemented CRYSTALS-Kyber, the first PQC algorithm standardized by NIST, on Xilinx Virtex-7 FPGA [13], [18]. Two versions of Kyber were coded in Verilog HDL. For both the unmasked and masked versions of the algorithm, they ran on the FPGA 4,000 times. Once 4,000 traces were collected, TVLA was applied. Given the results in Figures 7c and 7d, it could be found that the unmasked version had a leakage while the masked version had no leakage.

For the software side, the source code of CRYSTALS-Kyber from the PQM4 [19] library, which contains Cortex-M4 implementations of the PQC algorithms, was utilized. The interfaces were implemented by taking the sub-algorithms from the PQM4 implementation and placing them in the interface. The masked CRYSTALS-Kyber implementation was

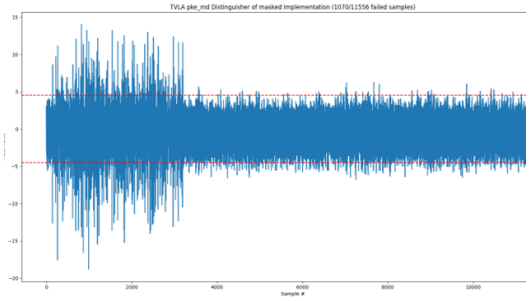


Fig. 8: TVLA result of the software masked Kyber.

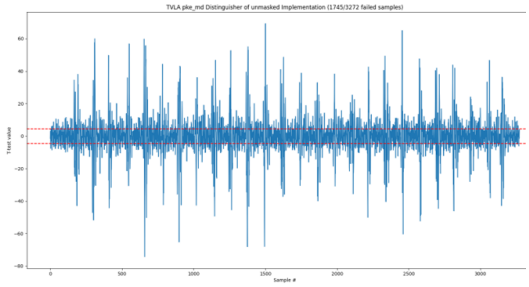


Fig. 9: TVLA result of the software unmasked Kyber.

taken from an open-sourced repository by Heinz et al. [20]. An example of the TVLA of the masked implementation can be found in Fig. 8. An example of the TVLA of the unmasked implementation can be found in Fig. 9. There is a larger reduction of leakage samples in the masked implementation compared with the unmasked implementation. 3rd round PQC algorithm Saber [21] was tested on the ChipWhisperer platform, but was not used for the infrastructure testing.

IV. Conclusion

In this work, we present a cloud-based infrastructure for post-quantum cryptography side-channel attack and analysis. This prototype contains the development of a set of tools to automate the process from accepting user inputs to returning testing reports. Results show that the infrastructure can successfully detect leakage of secrets from unmasked version of CRYSTALS-Kyber algorithm. This infrastructure can be used to rigorously test cryptographic algorithms that are integrated into the power electronics to ensure secure communication in power grid.

One of the limitations of this work is the use of the infrastructure's distinguishers and the inputs that are used for measuring. When looking at distinguishers, they usually cover a variable that can be composed of many other variables. For instance, the ciphertext in Kyber is composed of some secret variables and some public variables. It may be possible that the testbench accidentally measures the leakage of the public variable, which gives the false impression that there is leakage of the secret variable [22] that is being analyzed through

the TVLA. It may be the possible reason why Fig. 8 fails the TVLA. Possibly creating better crafted test vectors with relation to the distinguisher would help with this situation. Another issue of the prototype is that the algorithm implementations can run for so long that the oscilloscope does not have enough memory to capture the whole trace. For some software masked implementations, the overhead introduced into the runtime was large enough that the oscilloscope could not measure the specified point of interest. The oscilloscope used in the prototype has a limited sample buffer length and must follow the Nyquist-Shannon theorem, meaning there is a maximum number of samples that can be measured. This means that the infrastructure tool will first need to measure the runtime of the implementation to gauge if it could be measured or would require a more powerful oscilloscope (in terms of sample length).

Given the simplicity of the major system components, this infrastructure can be scaled up to service many users. The prototype currently contains a self-hosted Flask server, one sandboxed computer that operates the ChipWhisperer testbench, and another sandboxed computer that operates the oscilloscope testbench. To scale this to a larger size, it would require a larger server to help process the jobs, more sandbox computers to control testbenches and perform TVLA.

Some plans for this infrastructure involve fully prototyping the infrastructure tool. The prototype currently has both an automated capturing process and an automated TVLA process for the two distinguishers mentioned before. The current prototype also has the website up and running with a database along with scripts to help run the TVLA process. The work left in the prototype involves automating the whole end-to-end process from the user end all the way to the testbench and returning results. Some additional plans for the infrastructure tool involve adding more platforms to the list to allow users to have more freedom in their implementations. The infrastructure prototype currently has the STM32F4 and the Virtex-7 VC707 FPGA victim targets. There are also plans to add electromagnetic radiation analysis using near-field probes to capture leakage that can only be seen by those types of probes.

ACKNOWLEDGMENT

This work is supported in part by NSF Grant 2213738, and U.S. Department of Education Award P200A180019. The authors like to thank Tendayi Kamucheka for his help on the development of the infrastructure.

REFERENCES

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annual Symposium on Foundations of Computer Science (SFCS '94)*, Nov. 1994, pp. 124–134.

[2] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016, vol. 12.

[3] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM," in *Proceedings - 3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*. Institute of Electrical and Electronics Engineers Inc., jul 2018, pp. 353–367.

[4] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium : A Lattice-Based Digital Signature Scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 238–268, feb 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839

[5] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-fourier lattice-based compact signatures over ntru.(2018)," *Submission to the NIST PQC project*, vol. 3, 2018.

[6] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The sphincs+ signature framework," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2129–2146.

[7] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 81–88.

[8] D. Amiet, A. Curiger, L. Leuenberger, and P. Zbinden, "Defeating newhope with a single trace," Cryptology ePrint Archive, Report 2020/368, 2020, https://ia.cr/2020/368.

[9] P. Ravi, S. Sinha Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based pke and kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, p. 307–335, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8592

[10] M. J. Kannwischer, P. Pessl, and R. Primas, "Single-trace attacks on keccak," Cryptology ePrint Archive, Report 2020/371, 2020, https://ia.cr/2020/371.

[11] A. Abdulgadir, W. Diehl, and J.-P. Kaps, "An open-source platform for evaluation of hardware implementations of lightweight authenticated ciphers," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, 2019, pp. 1–5.

[12] C. O'Flynn and Z. D. Chen, "Chipwhisperer: An open-source platform for hardware embedded security research," in *Constructive Side-Channel Analysis and Secure Design*, E. Prouff, Ed. Cham: Springer International Publishing, 2014, pp. 243–260.

[13] T. Kamucheka, A. Nelson, D. Andrews, and M. Huang, "A masked pure-hardware implementation of kyber cryptographic algorithm," in *2022 International Conference on Field-Programmable Technology (ICFPT)*, 2022, pp. 1–1.

[14] B. J. Gilbert Goodwill, J. Jaffe, P. Rohatgi *et al.*, "A testing methodology for side-channel resistance validation," in *NIST non-invasive attack testing workshop*, vol. 7, 2011, p. 115–136.

[15] D. Heinz and T. Pöppelmann, "Combined fault and dpa protection for lattice-based cryptography," Cryptology ePrint Archive, Report 2021/101, 2021, urlhttps://eprint.iacr.org/2021/101.

[16] M. V. Beirendonck, J.-P. D'Anvers, and I. Verbauwhede, "Analysis and comparison of table-based arithmetic to boolean masking," Cryptology ePrint Archive, Report 2021/067, 2021, urlhttps://eprint.iacr.org/2021/067.

[17] M. J. Kannwischer, P. Schwabe, D. Stebila, and T. Wiggers, "Improving software quality in cryptography standardization projects," in *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022*. Los Alamitos, CA, USA: IEEE Computer Society, 2022, pp. 19–30. [Online]. Available: https://eprint.iacr.org/2022/337

[18] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of crystals-kyber pqc algorithm through resource reuse," *IEICE Electronics Express*, pp. 17–20 200 234, 2020.

[19] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "PQM4: Post-quantum crypto library for the ARM Cortex-M4," https://github.com/mupq/pqm4.

[20] D. Heinz, M. J. Kannwischer, G. Land, T. Pöppelmann, P. Schwabe, and D. Sprenkels, "First-order masked kyber on arm cortex-m4," Cryptology ePrint Archive, Paper 2022/058, 2022, https://eprint.iacr.org/2022/058. [Online]. Available: https://eprint.iacr.org/2022/058

[21] J.-P. D'Anvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber. proposal to nist pqc standardization, round2, 2019."

[22] M.-J. O. Saarinen, "Wip: Applicability of iso standard side-channel leakage tests to nist post-quantum cryptography," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2022, pp. 69–72.