Building Applications and Developing Digital Signature Devices based on the Falcon Post-Quantum Digital Signature Scheme

Nhu Quynh Luc
Academy of Cryptography Techniques, Vietnam
quynhln@actvn.edu.vn
(corresponding author)

Tat Thang Nguyen
National Agency of Cryptography and Information Security, Vietnam
thangnt@bcy.gov.vn

Duc Huy Quach
Academy of Cryptography Techniques, Vietnam
qdhuy2000gl@gmail.com

Toan Thanh Dao
University of Transport and Communications, Vietnam
daotoan@utc.edu.vn

Ngoc-Thao Pham
VNU University of Engineering and Technology, Vietnam
thao.pham@vnu.edu.vn

Received: 9 January 2023 | Revised: 23 January 2023 and 9 February 2023 | Accepted: 11 February 2023

ABSTRACT

Falcon is an efficient and secure postquantum signature scheme for services based on quantum computing. It employs the hash-and-sign approach in conjunction with the Gentry, Peikert, and Vaikuntanathan (GPV) framework on Number Theory Research Unit (NTRU) lattices. This study evaluated the operation procedure and the capacity to run the Falcon scheme using a key length of 1024 bits on different hardware and software platforms, such as personal computers and Raspberry Pi 4 and Windows, Ubuntu, and Android operating systems. The following results were obtained: file sizes ranged from 30 to 5449268KB, digital signature times ranged from 50 to 19500ms, and signature verification times ranged from 14 to 19000ms. The results show that the Falcon post-quantum signature scheme works stably and ensures execution speed on different platforms, similar to current digital signature schemes.

Keywords-post-quantum; signature; falcon; NTRU lattices; Raspberry Pi 4 Model B

I. INTRODUCTION

With the emergence of quantum computers, traditional cryptography is gradually losing its security elements, and becomes a threat to the security of asymmetric cryptosystems and digital signatures based on number theory in quantum computers, such as RSA, DSA, Diffie-Hellman, ElGamal, and elliptic curve variants [1-4]. Post-quantum cryptosystems must ensure their security properties even in the face of quantum computers [4-5]. The Falcon scheme is one of the post-quantum digital signature schemes nominated in the post-quantum cryptography competition that can be used on NIST quantum computer systems as of 2017 [1]. NIST has presented several cryptosystems that can meet the security requirements against quantum computing systems, known as post-quantum cryptography, such as NTRU, Rainbow, Classic McEliece, Falcon, etc. [1]. In particular, the Falcon digital signature is one of the schemes that has many security advantages against quantum computer systems [6-7].

The GPV framework theory for lattice-based digital signatures was presented in 2008 [2]. The design of the Falcon signature scheme was based on the lattice theory for the digital signature scheme [2]. This lattice theory is constructed by first initializing it with the NTRU lattice and the trapdoor sampler "Fourier Rapid Sampling" [8-9]. The complexity of the Falcon signature scheme is based on the openness of finding the Short Integer Solution (SIS) when solving the NTRU lattice problem [10]. This is a current open problem. This problem has been solved in cases where the boundary conditions of the equation are small, but the solution is still challenging when the boundary conditions of the equation are large, even with the help of a quantum computer.

This study investigated and assessed the design and implementation of the Falcon digital signature scheme for key generation, digital signature, and signature verification on hardware (Raspberry Pi 4 Model B [11-13]) and software (Windows, Ubuntu, Android), using experiments to assess its performance.

II. THE FALCON SCHEME

A. Mathematical Basis of the Falcon Post-Quantum Signature Scheme

In the Falcon digital signature scheme, the GPV framework theory is used to build a lattice-based digital signature scheme. The framework can be described as having the following components:

- The public key is a matrix with full rank $A \in Z_q^{n\times m}$, $m > n$, that creates a lattice Λ.
- The secret key is a matrix $B \in Z_q^{n\times m}$ that creates an orthogonal mesh $\Lambda_q^{\perp}$. Here, $\Lambda_q^{\perp}$ is the orthogonal grid symbol of the lattice Λ modulo q and the orthogonal mesh satisfies the following property: with every $x \in \Lambda$ and $y \in \Lambda_q^{\perp}$, then the condition $\langle x, y\rangle = 0 \bmod q$ is satisfied. Equivalently, the orthogonal rows of A and B satisfy:

$$B \times A^t = 0 \quad (1)$$

- Perform signature for message m: the signature form is a short integer value $s \in Z_q^m$, so that $sA^t = H(m)$, therein $H: \{0,1\}^* \to Z_q^n$ is a hash function. Then, for A, the validation of signature s is performed simply by checking if s is a short integer value that satisfies the condition $sA^t = H(m)$.
- Signature verification: The signature validation process is more complicated. At first, the user has to calculate the pre-image value $C_0 \in Z_q^m$ to satisfy $C_0A^t = H(m)$. This is entirely possible with linear algebra calculus tools as C_0 is not required to be short and $m \geq n$. Then, B is used to compute the orthogonal closing vector $v \in \Lambda_q^{\perp}$ close to C_0. The validity of the signature is determined by $s = c_0 - v$. When c_0 and v are close enough (small $c_0 - v$) then:

$$s\,A^t = c_0A^t - vA^t = c - 0 = H(m) \quad (2)$$

As a result, s is short. This shows that the Falcon signature scheme has the advantage that the signature must be short.

In the GPV frame, v is calculated based on the algorithm randomness generated in the algorithm variant to find the nearest plane corresponding to v [14]. As the algorithm to find the primitive nearest plane is vulnerable to an attack on the corresponding basis set of the secret corresponding to B, the schema is unsafe. However, this was improved when the algorithm was used with a given m and sampling s according to the demand distribution [15]. The spherical Gaussian distribution on the translated lattice is $C_0 + \Lambda_q^{\perp}$. This method was proven to not reveal information about B, and it was the first algorithm to use trapdoor sampling sets.

Afterward, choosing a cryptosystem for the GPV frame is a requirement. The Falcon post-quantum digital signature scheme used an NTRU lattice in addition to a ring structure. The purpose of this idea was to help reduce the size of public keys with computational complexity $O(n)$ and speed up the scheme by reducing the computational complexity to $O(n/\log n)$. In terms of the theory of lattice on rings, the NTRU lattice has been proven to be the smallest standard grid, i.e. the smallest set that has many good properties. The good cryptographic properties of the NTRU lattice on this polynomial ring are shown by the following property: The public key is a reminder of a simple (one-variable) polynomial on the ring of polynomial $h \in Z_q[x]$ whose largest degree is n-1. With the advantages of the NTRU lattice when applying such public key generation, the GPV framework used with the NTRU grid ensures the security of the Falcon scheme [9]. The NTRU grid is represented as:

$$\varphi = x^n + 1\,,\ (n = 2^k) \quad (3)$$

The secret key of NTRU is a set of four polynomials f, g, F, $G \in Z[x]/(\varphi)$, that satisfy:

$$fG - gF = q \bmod \varphi \quad (4)$$

where the polynomial f must be invertible modulo q. For the public key, the polynomial h can be calculated by:

$$h \leftarrow g \cdot f^{-1} \bmod q \quad (5)$$

The polynomial h is called the public key. Thus, for the Falcon scheme, the public key is the polynomial h and the secret key is a set of the four polynomials f, g, F, G.

B. Prove the Correctness of the Key Generation Process

The two matrices $\begin{bmatrix}1 & h\\ 0 & q\end{bmatrix}$ and $\begin{bmatrix}f & g\\ F & G\end{bmatrix}$ must be on the same lattice. In this case, if the polynomials f and g are produced with a sufficiently large entropy, then the generated public key h guarantees good pseudo-randomness [9]. However, in practice, even if f and g have relatively small entropy, it is still difficult to find the corresponding small polynomials f' and g' satisfying the condition $h = g' \cdot (f')^{-1} \bmod q$. This makes the NTRU lattice difficult to solve when the lattice is large enough, increasing complexity and ensuring its safety against quantum computers.

C. Implementation of the Falcon Post-Quantum Signature Scheme

The Falcon key generation, digital signature, and authentication are built on top of the GPV framework and implemented as follows:

- For the key generation, the public key is $A = [1 \mid h^*]$, which is equivalent to knowing the polynomial h.
- The secret key is: $B = \begin{bmatrix} g & -f \\ G & -F \end{bmatrix}$
- For the key validation, A and B are orthogonal through the expression: $B \times A^* = 0 \; mod \; q$.
- For the digital signature, the signature of the message m takes the form of a salt r along with a pair of polynomials (s_1, s_2) satisfying:

$$s_1 + s_2 h = H(r||\mathrm{m}) \quad (6)$$

Since s_1 is completely determined by m, r, and s_2, the signature is a pair (r, s_2).

D. Selecting a Set of Parameters to Ensure the Safety of the Falcon Scheme

The input parameters to the Falcon signature scheme are important to ensure a secure digital signature process. The scheme is built on the GPV framework on defining the sample for the trapdoor sampler. The inputs to the trapdoor sampler include matrix A, trapdoor function T, and objective value c. The output is a short vector s that satisfies:

$$s^t A = c \; mod \; q \quad (7)$$

Calculating this output value is equivalent to finding a vector $v \in \Lambda_q^{\perp}$ that has a value close enough to c_0. This shows that the tailgate sampler is important. These input parameter values are taken to ensure the quality of the trapdoor sampler based on efficient matrix calculations, and the "quality" of the sampler must be guaranteed: The shorter the vector s, i.e. the closer v is to c_0, the safer the sample.

E. Theoretical Security Assessment for the Falcon Post-Quantum Digital Signature Scheme

With theoretical safety criteria, the NIST has made several evaluations of current trapdoor samplers. Table I details the survey, analysis, and performance evaluation in terms of speed, outputs, and compatibility with the NTRU lattice.

TABLE I. COMPARE SAMPLING ALGORITHMS [1]

Sampling Algorithm	Speed	Output s short	NTRU lattice compatibility
[15]	No	Yes	No
[17]	Yes	No	Yes
[18]	Yes	Yes	No
[8]	Yes	Yes	Yes

In the implementation of the algorithm presented in [5], matrix B is taken as a trapdoor and then the algorithm generates vector s with normalized form $\|B\|_{GS}$. The process of generating a short vector s increases the security of the algorithm. This process has a computational complexity in time and space of approximately $O(m^2)$ [16].

The algorithm proposed in [17] is a version of the algorithm for finding the nearest plane at random. In [17], it was shown that this algorithm was equivalent to [15], and the output s was also a vector, but expressed in a normalized form $\|B\|_2$. But, as s is represented in its second normal form, the security will not be equal to [15]. In terms of complexity and processing time, this algorithm has time and space complexity of $O(m\ logm)$. The algorithm in [17] allows sampling the trapdoor from the trapdoor of A simply and efficiently. However, this algorithm is not compatible with the NTRU lattice and does not achieve a small enough initial vector s to correspond to the NTRU lattice built, according to the GPV framework [18]. The algorithm "fast Fourier transform for the nearest plane" proposed in [8] was a variant of the algorithm "find the nearest plane of Babai" with a lattice on a polynomial ring. In this algorithm, recursion is very similar to a fast Fourier transform, which is the reason for the algorithm's name. The algorithm was built based on the trapdoor sampling method with assurance according to the [15]. This shows that the algorithm in [8] works as efficiently as the algorithm in [17] and can be used with the NTRU lattice.

According to Table I, the "fast Fourier transforms for nearest plane" algorithm on the NTRU grid [8] is the most suitable for the objectives of this study. After generating the key according to the NTRU lattice, the polynomials f, g, F, G will be converted to a canonical form for use as a new secret key of the form $sk=(\hat{B},T)$. Matrix $\hat{B}$ is calculated according to:

$$\hat{B} = \begin{bmatrix} FFT(g) & -FFT(f) \\ FFT(G) & -FFT(F) \end{bmatrix} \quad (8)$$

The calculation of the falcon tree T is performed in 2 steps. At first, T-tree is calculated from $G \rightarrow \hat{B} \times \hat{B}^*$ as an unnormalized Falcon tree. Then, normalization is performed on T-tree, according to standard deviation σ. The key generated in this way ensures compactness and allows fast signature generation.

III. RESULTS AND DISCUSSION

A. Design and Build Falcon Post-Quantum Digital Signature Application on Windows, Ubuntu, and Android OS

The Falcon post-quantum digital signature module used in this study includes the following modules: quantum key generation (based on NTRU lattice combined with the falcon tree), Falcon digital signature (according to the GPV framework built on NTRU lattice with trapdoor sampling "Quick Fourier Transform Sampling"), and Falcon digital signature authentication. Figure 1 shows a working model of the Falcon post-quantum digital signature. On the first start of the Falcon, after entering the input file, the program generates a key according to the above algorithms if no standard set of keys has been generated or set before. The private part of this key will be used to digitally sign the file, and the digital signature of the original file along with the public key will be transmitted to the recipient so that he can verify it.

The program module was designed with 2 main interfaces: Figure 2 shows the interface for the digital signature process

according to the Falcon digital signature scheme to (a) sign and protect a file against post-quantum algorithms and (b) validate it. To sign a file, a user has to select it, type the filename to save the signed one, and press the "Sign" button. If the key exists, the program will immediately return the signing result. If not, it will generate a quantum key as described above. If the generation of the digital signature is successful, the signature will be printed along with the time elapsed for its generation. When authenticating a signed file, a user has to select it along with the public key of the signature and then click the "Verify" button. The program will automatically verify the signature, showing the appropriate message and the time elapsed for the validation procedure. The application was built in Python/Tkinter [19] and tested on Windows, Ubuntu, and Android, as shown in Figure 3.

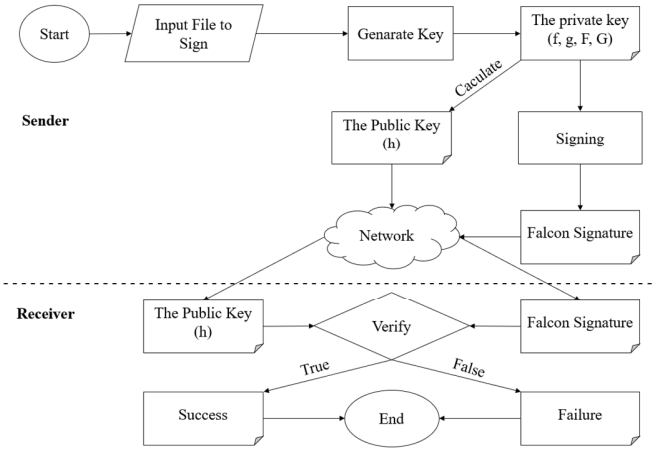


Fig. 1. Falcon Post-quantum digital signature operation model

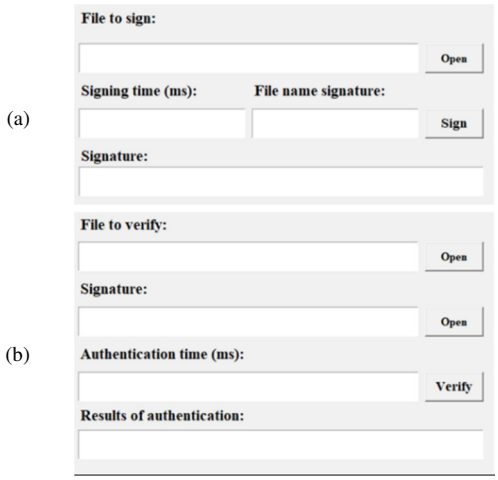


Fig. 2. Implemented (a) Falcon Post-Quantum digital signature and (b) signature authentication interfaces.

B. Development of Falcon Post-Quantum Digital Signature Device on Raspberry Pi 4 Model B Hardware Platform

In addition to evaluating the software, this study built a digital signature mechanism using a Raspberry Pi 4. The Falcon digital signature system was installed on a Raspberry Pi 4 Model B connected to a wifi router. Users accessed the system's IP address to utilize the service, as seen in Figure 4. To use the program, the system must be powered on, as seen in Figure 6(a), then access the system's IP address from the client and log in, as seen in Figure 6(b). After a successful authentication, the user can use the services using a pre-initialized key saved on the device, as seen in Figure 6(c). The user then can upload the file to the system, which will digitally sign it with a pre-initialized key and return the signed file to download.

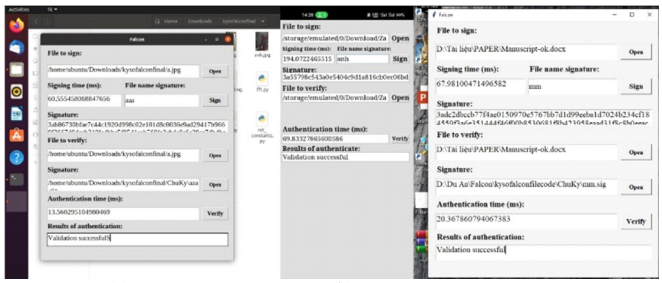


Fig. 3. Executing the application in (a) Ubuntu, (b) Android, and (c) Windows.

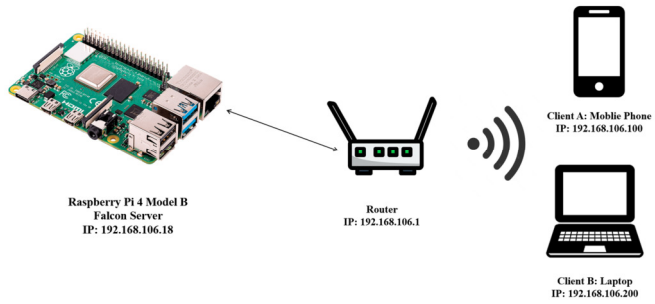


Fig. 4. Flowchart of Falcon digital signature system operation using a Raspberry Pi 4.

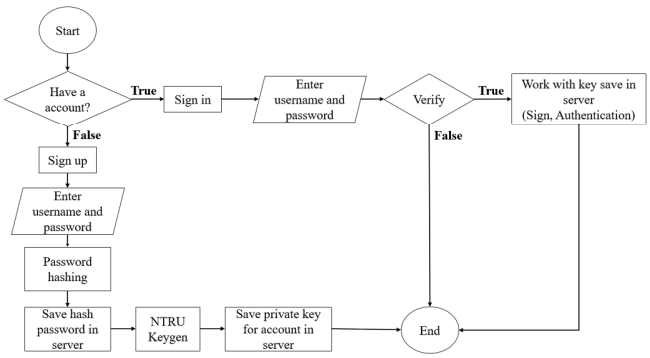


Fig. 5. Flowchart of key generation, key storage, and key use on the server.

This procedure uses four algorithms. The key generation algorithm is:

Require: Request the creation of a new account.

Ensure: a secret key sk, a public key pk.

1.Random ϕ = x1024 + 1

2.Calculate q = 121024 + 1

3.Compute NTRU polynomials f, g, F, G verifying: $fG - gF = q \bmod \phi$ (sk)

4.Calculate pk: $h \leftarrow g \cdot f^{-1} \bmod q$

5.Save sk, pk for this account in a yaml file.

The signing algorithm is detailed as:

Require: A message m, a secret key sk.

Ensure: A signature sig of m.

1.Random salt r

2.Hash the message m with salt r: $H(r||m)$

3.Repeat the signing procedure until finding a signature that is short enough (both the Euclidean norm and the byte length)

4.Return: a signature sig of m

The verification algorithm is:

Require: A signature sig of m, a public key pk, a message m'

Ensure: accept or not

1.Unpack the salt r and the short polynomial $s2$

2.Compute $s1$ and normalize its coefficients in $(-q/2, q/2]$

3.Check that the ($s1$, $s2$) is short

4.Check that $s1 + s2h = H(r||m')$

5.If all checks are passed, accept

The login verification algorithm is:

Require: Username, password

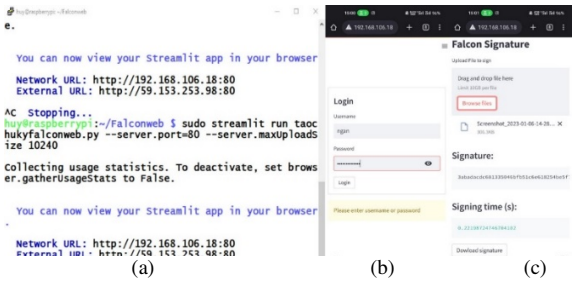
Ensure: accept or not

1.Check whether the username is in the database (yaml file)

2. Hash the password (hp')

3. Check that $hp' = hp$ (hp is the hash password for username in the yaml file)

4.If all checks are passed, accept



(a) (b) (c)

Fig. 6. (a) Raspberry Pi 4 System startup, (b) login, and (c) digital signature interface.

To evaluate the performance of the system, different file types were digitally signed on these platforms with a key length of 1024 bits. Table II shows the results obtained for the Falcon post-quantum signature process and verification. These results show that the execution time for the Falcon digital signature system on various platforms demonstrates the program's efficacy and stability. Digital signing takes 50–19500ms, signature authentication takes 14–19000ms, and the signature size is 1.28KB for file sizes ranging from 30 to 5,449,268KB, on different hardware (personal computer and Raspberry Pi 4) and software (Windows, Ubuntu, and Android). It is clear that the digital signature speed is related to the size of the file but always yields the same signature file size. The Falcon post-quantum cryptography can be used securely for quantum computers, as it has shorter execution times than the Rainbow post-quantum cryptosystem on the same machine [1]. This can be explained by Rainbow ($O(n^3)$) having more computational complexity than Falcon ($O(n/\log n)$).

TABLE II. FALCON SIGNATURE AND VALIDATION TIMES ACROSS PLATFORMS AND RAINBOW

File type	Capacity (KB)	Signature time (ms)	Authentication time (ms)	Signature size (KB)
Windows 10 [16Gb RAM – 2.3Ghz (8 CPUs)]				
jpg	124	50.896	14.994	1.28
exe	636,133	2314.817	2230.037	1.28
rar	2,977,754	11073.385	10615.610	1.28
iso	5,449,268	19448.226	19014.150	1.28
Ubuntu 20.04.4 – VMWare [16Gb RAM – 2.3Ghz (2 CPUs)]				
jpg	127	57.470	13.882	1.28
rar	40,803	208.493	164.825	1.28
exe	636,133	2481.803	2446.245	1.28
deb	3,335,876	12502.191	11989.483	1.28
Android [6Gb RAM – Snapdragon 695 5G]				
pdf	6758.4	164.661	99.195	1.28
rar	39,843	294.353	229.441	1.28
apk	357,808	1445.067	1350.679	1.28
mp4	1,184,891	4783.171	4470.747	1.28
Raspberry Pi 4 Model B [4Gb RAM - ARM Cortex-A72]				
File type	**Capacity (KB)**		**Signature time (ms)**	**Signature size (KB)**
jpg	4,855		246.922	1.28
mov	102,363		911.44	1.28
rar	134,607		1128.04	1.28
exe	636,133		4632.76	1.28
Rainbow - Windows 10 [16Gb RAM – 2.3Ghz (8 CPUs)] [1]				
File type	**Capacity (word)**		**Signature time (ms)**	**Authentication time (ms)**
Text	500		108.3	38.8
Text	1000		46.4	37.4
Text	10000		20.7	28.9

The Fortify Static Code Analyzer toolkit v.22.1.0.0166 was used to validate the code structure, showing that the code was created securely. These results show that this application has reasonably fast post-quantum digital sign and verification speeds, which meet the current user requirements.

IV. CONCLUSION

This study investigated the mathematical assessment and security guarantees for the Falcon post-quantum digital signature system and tested its schema's key generation, and digital signature and validation processes. The results obtained with a key length of 1024 bits for the Falcon schema on both hardware and software (running on Windows, Ubuntu, and Android platforms) for file sizes between 30-5,449,268KB.

The time to perform a digital signature was approximately 50-19,500ms, the time to perform signature authentication was approximately 14-19,000ms, and the signature size was 1.28KB. These results demonstrate that the Falcon post-quantum digital signature method has steady performance and guarantees the same execution speed as existing digital signature systems on a variety of platforms. Future work should investigate the use of the Falcon post-quantum digital signature on more hardware platforms and its integration into commercial applications.

ACKNOWLEDGMENT

The authors acknowledge the Academy of Cryptography Techniques and the Minister of Education and Training (MOET) for supporting this work under grant numbers B2022-GHA-09 and B2022-GHA-10.

REFERENCES

[1] G. Alagic *et al.*, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process," National Institute of Standards and Technology, NIST Internal or Interagency Report (NISTIR) 8413, Sep. 2022. https://doi.org/10.6028/NIST.IR.8413-upd1.

[2] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, Victoria, Canada, Feb. 2008, pp. 197–206, https://doi.org/10.1145/1374376.1374407.

[3] U. Iftikhar, K. Asrar, M. Waqas, and S. A. Ali, "Evaluating the Performance Parameters of Cryptographic Algorithms for IOT-based Devices," *Engineering, Technology & Applied Science Research*, vol. 11, no. 6, pp. 7867–7874, Dec. 2021, https://doi.org/10.48084/etasr.4263.

[4] R. Bhat, N. R. Sunitha, and S. S. Iyengar, "A probabilistic public key encryption switching scheme for secure cloud storage," *International Journal of Information Technology*, Sep. 2022, https://doi.org/10.1007/s41870-022-01084-8.

[5] N. M. Mukhammadovich and A. R. Djuraevich, "Working with cryptographic key information," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 1, pp. 911–919, Feb. 2023, https://doi.org/10.11591/ijece.v13i1.pp911-919.

[6] H. M. Bahig, A. Alghadhban, M. A. Mahdi, K. A. Alutaibi, and H. M. Bahig, "Speeding up the Multiplication Algorithm for Large Integers," *Engineering, Technology & Applied Science Research*, vol. 10, no. 6, pp. 6533–6541, Dec. 2020, https://doi.org/10.48084/etasr.3932.

[7] M. F. Hyder, S. Tooba, and Waseemullah, "Performance Evaluation of RSA-based Secure Cloud Storage Protocol using OpenStack," *Engineering, Technology & Applied Science Research*, vol. 11, no. 4, pp. 7321–7325, Aug. 2021, https://doi.org/10.48084/etasr.4220.

[8] L. Ducas and T. Prest, "Fast Fourier Orthogonalization," in *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*, Waterloo, Canada, Apr. 2016, pp. 191–198, https://doi.org/10.1145/2930889.2930923.

[9] D. Stehlé and R. Steinfeld, "Making NTRU as Secure as Worst-Case Problems over Ideal Lattices," in *Advances in Cryptology – EUROCRYPT 2011*, Tallinn, Estonia, 2011, pp. 27–47, https://doi.org/10.1007/978-3-642-20465-4_4.

[10] D. Das, V. Saraswat, and K. Basu, "Lattice signatures using NTRU on the hardness of worst-case ideal lattice problems," *IET Information Security*, vol. 14, no. 5, pp. 496–504, 2020, https://doi.org/10.1049/iet-ifs.2019.0580.

[11] W. Donat, Learn Raspberry Pi Programming with Python: Learn to Program on the World's Most Popular Tiny Computer, 2nd ed. O'Reilly Media Inc., 2018.

[12] "Datasheet Raspberry Pi Model B." https://datasheets.raspberrypi.com/.

[13] Edwar Jacinto Gomez; Caterinne Perilla Gutierrez; Lina Uyasaba Murillo, "Hardware based cryptography: technological advances for applications in Colombia using embedded systems," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 508–517, Feb. 2021, https://doi.org/10.11591/ijece.v11i1.pp508-517.

[14] G. McGuire and O. Robinson, "Lattice Sieving in Three Dimensions for Discrete Log in Medium Characteristic," *Journal of Mathematical Cryptology*, vol. 15, no. 1, pp. 223–236, Jan. 2021, https://doi.org/10.1515/jmc-2020-0008.

[15] P. Klein, "Finding the closest lattice vector when it's unusually close," in *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, San Francisco, CA, USA, Oct. 2000, pp. 937–941.

[16] P. Q. Nguyen and T. Vidick, "Sieve algorithms for the shortest vector problem are practical," *Journal of Mathematical Cryptology*, vol. 2, no. 2, pp. 181–207, Jul. 2008, https://doi.org/10.1515/JMC.2008.009.

[17] C. Peikert, "An Efficient and Parallel Gaussian Sampler for Lattices," in *Advances in Cryptology – CRYPTO 2010*, Santa Barbara, CA, USA, 2010, pp. 80–97, https://doi.org/10.1007/978-3-642-14623-7_5.

[18] D. Micciancio and C. Peikert, "Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller," in *Advances in Cryptology – EUROCRYPT 2012*, Cambridge, UK, 2012, pp. 700–718, https://doi.org/10.1007/978-3-642-29011-4_41.

[19] Y. Chen, N. Genise, and P. Mukherjee, "Approximate Trapdoors for Lattices and Smaller Hash-and-Sign Signatures," in *Advances in Cryptology – ASIACRYPT 2019*, Kobe, Japan, 2019, pp. 3–32, https://doi.org/10.1007/978-3-030-34618-8_1.

[20] T. Weber, R. Georgii, and P. Böni, "Takin: An open-source software for experiment planning, visualisation, and data analysis," *SoftwareX*, vol. 5, pp. 121–126, Jan. 2016, https://doi.org/10.1016/j.softx.2016.06.002.