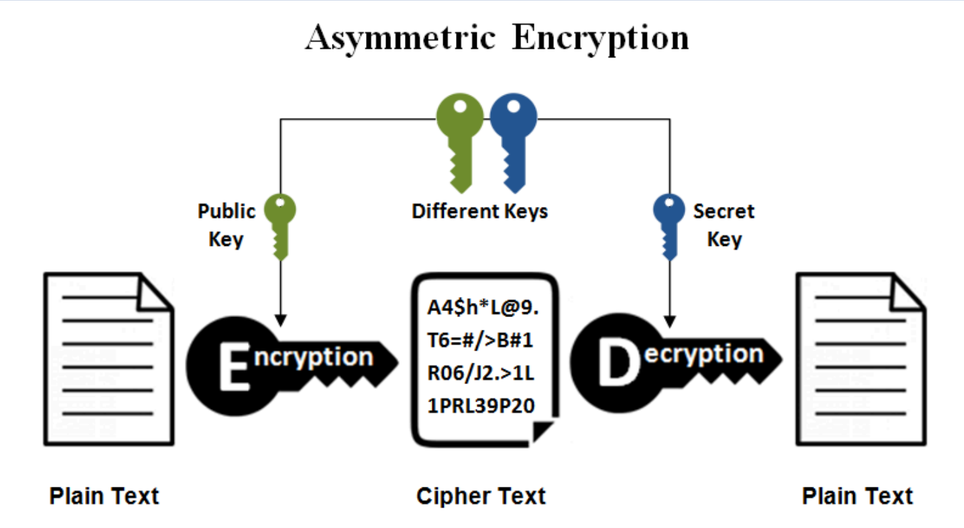
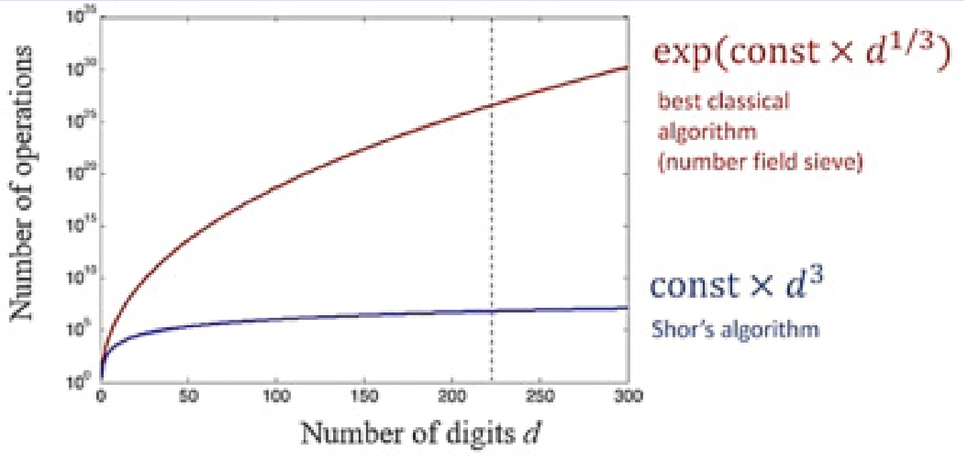
Post-quantum cryptography Algorithm's standardization and performance analysis

MANISH KUMAR
DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS, M S RAMAIAH INSTITUTE OF TECHNOLOGY, BANGALORE, 54, INDIA
RECEIVED 2 APRIL 2022, REVISED 4 AUGUST 2022, ACCEPTED 4 AUGUST 2022, AVAILABLE ONLINE 18 AUGUST 2022, VERSION OF RECORD 23 AUGUST 2022.

Introduction





Modern Cryptography

The concept of a “computationally secure scheme” is that it is theoretically possible to crack such a system, but practically impossible to do so.

Quantum computer

A practical quantum computer with sufficient numbers of qubits will be able to break all the modern public-key cryptographic systems

Shor’s algorithm

Successful implementation of Shor's algorithm could easily break the most widely used cryptographic algorithm such as RSA, Elliptic Curve Cryptography (ECC), and Diffie-Hellman.

LA

Many countries and standard organizations have taken an initiative in the design, development, testing, and migration strategy for quantum-safe algorithms

- National Institute of Standards and Technology (NIST)
- The European Telecommunications Standards Institute (ETSI)
- The Internet Engineering Task Force (IETF)
- The American National Standards Institute (ANSI)
- Federal Office for Information Security (BSI)

Main objective

Comparative Analysis

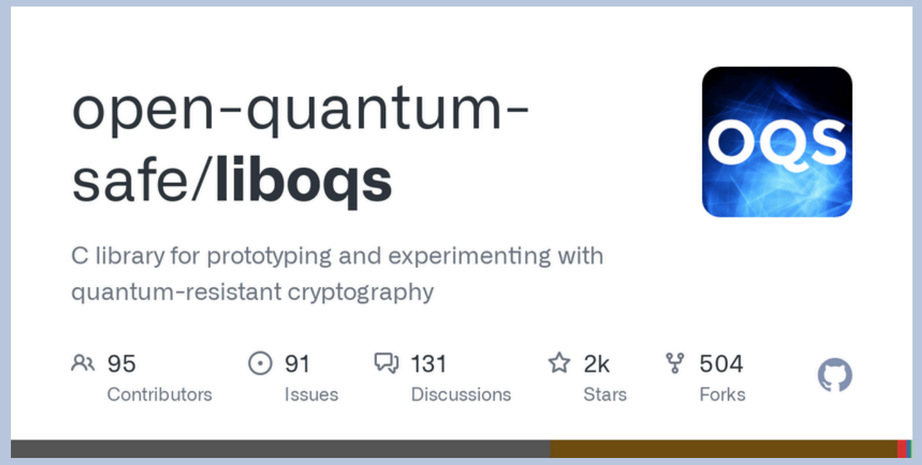
However, not much information is available in the public domain except for the work done by NIST. Hence, decided to do a comparative analysis of 7 finalists and 8 alternate quantum-resistant algorithms announced by the NIST during the 3rd round in the year 2020

Implementation

to assess the implementation feasibility of the prospective algorithms based on their CPU cycle and memory utilization.

LA

Technical methods



The performance analysis of the algorithms is done using the Open Quantum Safe (OQS) project. It is a project, developing and prototyping quantum-resistant cryptography algorithms The OQS also provides benchmarking data for various quantum-resistant algorithms which are used in this paper for the comparative analysis of the algorithm's runtime behavior and memory consumption. The runtime behavior and memory consumption data of the algorithms are collected based on the execution of the algorithms on Amazon Web Service (AWS) with CPU Model Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50 GHz.

Types

Code-based cryptography

It's based on error-correcting codes, which are used to fix mistakes in data transmission.

$$codeword = [1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1]$$

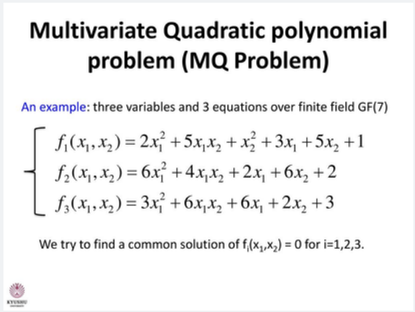
$$received\ codeword = [1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0]$$

$$syndrome = [1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0] \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0\ \ 0\ \ 1]$$

$$received\ codeword = [1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1]$$

Multivariate Quadratic Equations-Based Cryptography

It's based on solving systems of equations with many variables

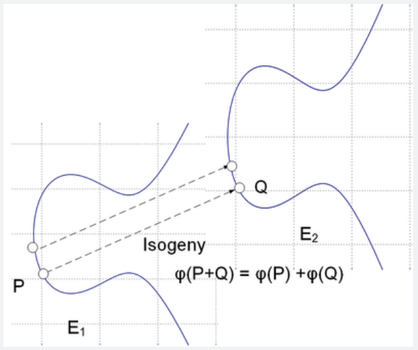


Hash-Based Cryptography

It's based on hash functions, taking any input (e.g., a message) and turning it into a fixed-size string of random-looking characters.

Isogeny-Based Cryptography

It's based on elliptic curves. Isogenies are like maps that transform one elliptic curve into another. The security comes from the difficulty of figuring out which map was used.



Lattice-Based Cryptography

It's based on lattices, which are like grids of points in multi-dimensional space. The security comes from the difficulty of finding the shortest path between points in this grid

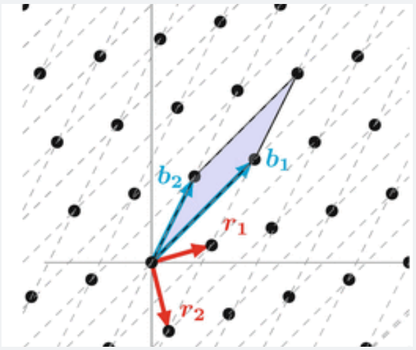


Table 11
Isogeny-based quantum-safe cryptographic algorithms.

Sl. No.	Algorithm Name	Current Status	Open-Source C Library for Quantum-Safe Cryptographic Algorithms (liboqs)				Usage
			NIST Level	Algorithms Name	Public Key Size (bytes)	Private Key Size (bytes)	
1	SIKE	Alternates	1	SIDH-p434	330	28	Key Encapsulation Mechanisms
				SIDH-p434-compressed	197	28	
				SIKE-p434	330	374	
				SIKE-p434-compressed	197	350	
			2	SIDH-p503	378	32	
				SIDH-p503-compressed	225	32	
				SIKE-p503	378	434	
				SIKE-p503-compressed	225	407	
			3	SIDH-p610	462	39	
				SIDH-p610-compressed	274	39	
				SIKE-p610	462	524	
				SIKE-p610-compressed	274	491	
			5	SIDH-p75	564	48	
				SIDH-p751-compressed	335	48	
				SIKE-p751	564	644	
				SIKE-p751-compressed	335	602	

Table 12
Runtime analysis of open quantum safe isogeny based cryptographic algorithms (Key encapsulation mechanisms).

Algorithm	Keygen/s	Keygen (cycles)	Encaps/s	Encaps (cycles)	Decaps/s	Decaps (cycles)
SIDH-p434 (x86_64)	109.63	22805106	53.61	46626958	135.2	18488925
SIDH-p434-compressed (x86_64)	57.09	43,797,293	38.69	64616727	119.7	20885902
SIDH-p503 (x86_64)	318.89	7840371	155.4	16,085,706	391.2	6389639
SIDH-p503-compressed (x86_64)	166.44	15020665	111.7	22385569	330.9	7,555,453
SIDH-p610 (x86_64)	30.91	80887652	15.93	156909224	37.1	67391175
SIDH-p610-compressed (x86_64)	18.88	132,403,975	13.28	188156935	36.07	69302679
SIDH-p751 (x86_64)	104.79	23,858,838	50.17	49,829,353	127.4	19629441
SIDH-p751-compressed (x86_64)	55.93	44696733	36.57	68372097	114.3	21,873,987
SIKE-p434 (x86_64)	98.8	25300016	60.57	41,275,416	56.52	44237605
SIKE-p434-compressed (x86_64)	57.71	43322569	36.58	68330870	52.79	47,358,432
SIKE-p503 (x86_64)	271.82	9197872	175	14283364	163	15,339,004
SIKE-p503-compressed (x86_64)	165.89	15,070,206	112.1	22309024	154.3	16200750
SIKE-p610 (x86_64)	30.84	81071205	16.84	148,430,862	16.75	149271935
SIKE-p610-compressed (x86_64)	18.99	131620069	13.28	188252832	16.99	147,137,618
SIKE-p751 (x86_64)	92.91	26906560	57.49	43,480,834	53.45	46777582
SIKE-p751-compressed (x86_64)	55.8	44,799,410	36.53	68,434,047	47.73	52,377,374

Table 13
Memory Use Analysis (bytes per algorithm) of Open Quantum Safe Isogeny Based Cryptographic Algorithms (Key Encapsulation Mechanisms).

Algorithm	Keygen (maxHeap)	Keygen (maxStack)	Encaps (maxHeap)	Encaps (maxStack)	Decaps (maxHeap)	Decaps (maxStack)
SIDH-p434 (x86_64)	4534	9928	5002	10,376	5084	9576
SIDH-p434-compressed (x86_64)	4401	78,248	4736	57,624	4818	9576
SIDH-p503 (x86_64)	9234	2192	9738	2320	9864	2320
SIDH-p503-compressed (x86_64)	4433	92,456	4816	63,840	9558	2304
SIDH-p610 (x86_64)	4677	14,520	5332	15,112	5447	14,008
SIDH-p610-compressed (x86_64)	4489	146,840	4956	104,072	5071	15,352
SIDH-p751 (x86_64)	4788	11,416	5588	12,216	5728	10,440
SIDH-p751-compressed (x86_64)	4559	196,440	5130	123,944	5270	10,760
SIKE-p434 (x86_64)	4880	10,152	5242	10,504	5258	11,192
SIKE-p434-compressed (x86_64)	4723	78,248	4975	58,072	4991	15,912
SIKE-p503 (x86_64)	9636	2192	10,062	2832	5438	7768
SIKE-p503-compressed (x86_64)	4808	92,456	5112	64,368	5136	13,784
SIKE-p610 (x86_64)	5162	14,856	5672	15,320	5696	16,248
SIKE-p610-compressed (x86_64)	4941	146,840	5301	104,712	5325	25,144
SIKE-p751 (x86_64)	5384	12,200	6012	12,376	6044	12,792
SIKE-p751-compressed (x86_64)	5113	196,456	5555	124,696	5587	22,440

Evaluation

- Key Size (bytes)
- Keygen/s
- Keygens(cycles)
- Encaps/s
- Encaps(cycles)
- Decaps/s
- Decaps(cycles)
- Keygen(maxHeap)
- Keygen(maxStack)
- Encaps(maxHeap)
- Encaps(maxStack)
- Decaps(maxHeap)
- Decaps(maxStack)
- NIST LEVEL 1(minimal),2(Intermediate),3(high) 4,5(military, classified)

NIST Level	Algorithms Name	Public Key Size (bytes)	Private Key Size (bytes)
1	Classic-McEliece-348,864	261,120	6452
	Classic-McEliece-348864f	261,120	6452
3	Classic-McEliece-460,896	524,160	13,568
	Classic-McEliece-460896f	524,160	13,568
5	Classic-McEliece-6688,128	1,044,992	13,892
	Classic-McEliece-6688128f	1,044,992	13,892

Algorithm	**Keygen (maxHeap)**	**Keygen (maxStack)**	**Encaps (maxHeap)**	**Encaps (maxStack)**	**Decaps (maxHeap)**	**Decaps (maxStack)**
BIKE-L1 (x86_64)	11,420	90,552	12,545	25,720	12,577	73,464
BIKE-L3 (x86_64)	17,844	179,448	20,511	50,296	20,543	144,952
Classic-McEliece-348,864 (x86_64)	271,748	2,205,032	276,556	3824	271,940	43,232
Classic-McEliece-348864f (x86_64)	271,748	2,205,032	276,556	3824	271,940	43,232
Classic-McEliece-460,896 (x86_64)	541,904	4,768,360	546,772	6528	542,156	80,256
Classic-McEliece-460896f (x86_64)	541,904	4,768,360	546,772	6528	542,156	80,256

- Offer strong security but have large public key sizes and high computational costs.

CODE-BASED ALGORITHMS

NIST Level	Algorithms Name	Public Key Size (bytes)	Private Key Size (bytes)
1	Rainbow-I-Classic	161,600	103,648
	Rainbow-I-Circumzenithal	60,192	103,648
	Rainbow-I-Compressed	60,192	64
3	Rainbow–III–Classic	882,080	626,048
	Rainbow–III–Circumzenithal	264,608	626,048
	Rainbow–III–Compressed	264,608	64
5	Rainbow-V-Classic	1,930,600	1,408,736
	Rainbow-V-Circumzenithal	536,136	1,408,736

Algorithm	Keypair/s	Keypair (cycles)	Sign/s	Sign (cycles)	Verify/s	Verify (cycles)
Rainbow-I-Circumzenithal (x86_64)	6.06	412,509,871	530.3	4,714,048	445.7	5,609,619
Rainbow-I-Classic (x86_64)	6.77	369,413,745	506	4,940,751	500.7	4,993,923
Rainbow-I-Compressed (x86_64)	6.02	415,229,320	13.77	181,591,225	445.9	5,607,001
Rainbow–III–Circumzenithal (x86_64)	0.43	1,535,117,154	54.95	45,496,023	49.06	50,968,436
Rainbow–III–Classic (x86_64)	0.49	815,304,452	54.82	45,599,192	50.58	49,426,107
Rainbow–III–Compressed (x86_64)	0.43	1,526,552,696	0.92	2,709,509,453	49.2	50,814,366
Rainbow-V-Circumzenithal (x86_64)	0.15	3,612,623,138	24.87	100,519,687	22.36	111,796,728
Rainbow-V-Classic (x86_64)	0.17	1,503,271,522	23.78	105,150,930	24.24	103,104,686

- Provide robust security but are resource-intensive in terms of both computation and memory

MULTIVARIATE QUADRATIC EQUATIONS-BASED ALGORITHMS

NIST Level	Algorithms Name	Public Key Size (bytes)	Private Key Size (bytes)
1	SPHINCS+-Haraka-128f-robust	32	64
	SPHINCS+-Haraka-128f-simple	32	64
	SPHINCS+-Haraka-128s-robust	32	64
	SPHINCS+-Haraka-128s-simple	32	64
	SPHINCS+-SHA256-128f-robust	32	64
	SPHINCS+-SHA256-128f-simple	32	64

Algorithm	Keypair/s	Keypair (cycles)	Sign/s	Sign (cycles)	Verify/s	Verify (cycles)
SPHINCS+-Haraka-128f-robust (x86_64)	3305.33	756,217	130.25	19193087	1841.3	1,357,668
SPHINCS+-Haraka-128f-simple (x86_64)	3021.67	827,241	126.75	19720897	2389.7	1046180
SPHINCS+-Haraka-128s-robust (x86_64)	49.8	50,199,666	6.26	399193532	5109	489,226
SPHINCS+-Haraka-128s-simple (x86_64)	53.95	46336950	6.88	363,156,129	7042	354,926
SPHINCS+-Haraka-192f-robust (x86_64)	2491	1003522	78.92	31672906	1248	2,003,318
SPHINCS+-Haraka-192f-simple (x86_64)	2850	877,072	99.34	25163368	1754.7	1424860

- Best suited for digital signatures with relatively small key sizes but can be computationally expensive.

HASH-BASED ALGORITHMS

NIST Level	Algorithms Name	Public Key Size (bytes)	Private Key Size (bytes)
1	SIDH-p434	330	28
	SIDH-p434-compressed	197	28
	SIKE-p434	330	374
	SIKE-p434-compressed	197	350
2	SIDH-p503	378	32
	SIDH-p503-compressed	225	32
	SIKE-p503	378	434
	SIKE-p503-compressed	225	407
3	SIDH-p610	462	39
	SIDH-p610-compressed	274	39
	SIKE-p610	462	524
	SIKE-p610-	274	491

Algorithm	Keygen/s	Keygen (cycles)	Encaps/s	Encaps (cycles)	Decaps/s	Decaps (cycles)
SIDH-p434 (x86_64)	109.63	22805106	53.61	46626958	135.2	18488925
SIDH-p434-compressed (x86_64)	57.09	43,797,293	38.69	64616727	119.7	20885902
SIDH-p503 (x86_64)	318.89	7840371	155.4	16,085,706	391.2	6389639
SIDH-p503-compressed (x86_64)	166.44	15020665	111.7	22385569	330.9	7,555,453
SIDH-p610 (x86_64)	30.91	80887652	15.93	156909224	37.1	67391175
SIDH-p610-compressed (x86_64)	18.88	132,403,975	13.28	188156935	36.07	69302679
SIDH-p751 (x86_64)	104.79	23,858,838	50.17	49,829,353	127.4	19629441
SIDH-p751-compressed (x86_64)	55.93	44696733	36.57	68372097	114.3	21,873,987
SIKE-p434 (x86_64)	98.8	25300016	60.57	41,275,416	56.52	44237605
SIKE-p434-compressed (x86_64)	57.71	43322569	36.58	68330870	52.79	47,358,432

- Provide compact key sizes but are computationally expensive, making them less efficient for some applications

ISOGENY-BASED ALGORITHMS

NIST Level	Algorithms Name	Public Key Size (bytes)	Private Key Size (bytes)
1	Kyber512	800	1632
	Kyber512-90s	800	1632
3	Kyber768	1184	2400
	Kyber768-90s	1184	2400
5	Kyber1024	1568	3168
	Kyber1024-90s	1568	3168
1	NTRU-HPS-2048-509	699	935
3	NTRU-HPS-2048-677	930	1234
	NTRU-HRSS-701	1138	1450
5	NTRU-HPS-4096-821	1230	1590
1	LightSaber-KEM	672	1568
3	Saber-KEM	992	2304

Algorithm	Keygen/s	Keygen (cycles)	Encaps/s	Encaps (cycles)	Decaps/s	Decaps (cycles)
FireSaber-KEM (x86_64)	21419.7	116,577	20332.3	122,807	21,608	115,601
FrodoKEM-1344-AES (x86_64)	589	4,244,465	463.18	5,397,844	475.17	5,261,594
FrodoKEM-1344-SHAKE (x86_64)	200.07	12,494,691	192.33	12,999,407	192.87	12962247
FrodoKEM-640-AES (x86_64)	2179	1147030	1578	1584240	1653	1512386
FrodoKEM-640-SHAKE (x86_64)	794	3,148,801	669.11	3735711	719.76	3473037
FrodoKEM-976-AES (x86_64)	1041.32	2400233	783.74	3189368	816	3,063,901
FrodoKEM-976-SHAKE (x86_64)	371.42	6,731,021	340.22	7348775	341.66	7316407
Kyber1024 (x86_64)	35,936	69,446	31185.3	80,016	41011.33	60,859
Kyber1024-90s (x86_64)	51866.3	48,061	45,639	54,625	67,317	37,038
Kyber512 (x86_64)	67,526	36,882	58398.7	42,674	92577.67	26,910
Kyber512-90s (x86_64)	86,699	28,696	81124.7	30,692	135048.3	18,406
Kyber768 (x86_64)	46470.7	53,657	42638.7	58,493	59629.33	41,825
Kyber768-90s (x86_64)	66,675	37,353	60583.7	41,115	96815.67	25,724

- Offer a good balance between security, key size, and performance, making them suitable for key encapsulation mechanisms.

LATTICE-BASED ALGORITHMS

Key findings

CPU Cycles

Performance analysis of various quantum-safe algorithms shows that in general, quantum-safe algorithms require many CPU cycles for basic operations

Key size

The algorithms also require large to a very large public key and private key sizes.

Memory

The runtime memory consumption of these algorithms is very high compared to the classical cryptographic algorithms.

KEM Selected for Standardization

Overall performance of KYBER in software, hardware, and hybrid settings are excellent.

Signatures Selected for Standardization

- CRYSTALS-Dilithium is Highly efficient and relatively simple in implementation.
- Falcon
  Because of its low bandwidth, it is a preferred choice for certain applications.
- SPHINCS+
  security seems solid and is based on an different set of assumptions than those of other

LA

CONCLUSION AND FUTURE WORK

- The exponential growth in quantum computer technology's development shows that the storm is approaching very fast. We need to migrate to post-quantum cryptographic algorithms at the earliest
- Additional four algorithms have been considered for advancement and 4th round of evaluation. The selected algorithm under consideration for the advancement and further evaluation in the 4th round needs to be upgraded on various performance factors to satisfy the implementation feasibility.
- Waiting for more information on public domain