Exploring NIST LWC/PQC Synergy with R5Sneik

How SNEIK 1.1 Algorithms were Designed to Support Round5

Markku-Juhani O. Saarinen*

PQShield Ltd.
Prama House, 267 Banbury Road
Oxford OX2 7HT, United Kingdom
mjos@pqshield.com

Abstract. Most NIST Post-Quantum Cryptography (PQC) candidate algorithms use symmetric primitives internally for various purposes such as "seed expansion" and CPA to CCA transforms. Such auxiliary symmetric operations constituted only a fraction of total execution time of traditional RSA and ECC algorithms, but with faster lattice algorithms the impact of symmetric algorithm characteristics can be very significant. A choice to use a specific PQC algorithm implies that its internal symmetric components must also be implemented on all target platforms. This can be problematic for lightweight, embedded (IoT), and hardware implementations. It has been widely observed that current NIST-approved symmetric components (AES, GCM, SHA, SHAKE) form a major bottleneck on embedded and hardware implementation footprint and performance for many of the most efficient NIST PQC proposals. Meanwhile, a separate NIST effort is ongoing to standardize lightweight symmetric cryptography (LWC). Therefore it makes sense to explore which NIST LWC candidates are able to efficiently support internals of post-quantum asymmetric cryptography. We discuss R5Sneik, a variant of Round5 that internally uses SNEIK 1.1 permutation-based primitives instead of SHAKE and AES-GCM. The SNEIK family includes parameter selections specifically designed to support lattice cryptography. R5Sneik is up to 40% faster than Round5 for some parameter sets on ARM Cortex M4, and has substantially smaller implementation footprint. We introduce the concept of a fast Entropy Distribution Function (EDF), a lightweight diffuser that we expect to have sufficient security properties for lattice seed expansion and many types of sampling, but not for plain encryption or hashing. The same SNEIK 1.1 permutation core (but with a different number of rounds) can also be used to replace AES-GCM as an AEAD when building lightweight cryptographic protocols, halving typical flash footprint on Cortex M4, while boosting performance.

Keywords: Post-Quantum Cryptography · Lightweight cryptography · Permutation-Based Cryptography · Round5 · SNEIK · R5Sneik · Blinker Protocol · Cortex M4

1 Introduction

NIST is currently running two separate cryptographic standardization projects, one on Post-Quantum Cryptography (PQC)[1], and another one on Lightweight Cryptography (LWC)[2].

[*] This work reuses parts of text written *by the author* for the Round5 PQC and SNEIK LWC proposals.

[1] Post-Quantum Cryptography Project: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography

[2] Lightweight Cryptography Project: https://csrc.nist.gov/Projects/Lightweight-Cryptography

Version 20190718151900

Both of these projects have issued calls for algorithm nominations and have received a large number of proposals, most of which are original designs. Both projects proceed in a number of stages where approximately half of the candidates are eliminated based on public cryptanalysis and broad technical criteria such as performance and implementation features.

The NIST PQC project seeks to create standards for quantum-resistant (asymmetric) signature, public key encryption, and key encapsulation (KEM) algorithms. This project issued its call in December 2016 and is currently in its 2nd or "semifinal" stage[AASA+19] with 26 candidate algorithms. Some of these algorithms will eventually be used to replace current RSA and Elliptic Curve standards, which are vulnerable to attacks by quantum computers [CNS15].

Meanwhile the NIST LWC project seeks to create standards for lightweight (low power, small footprint) symmetric primitives for Authenticated Encryption with Associated Data (AEAD) and cryptographic hashes. The LWC project issued its official call in August 2018 after a multi-year initial research and requirement definition phase. The proposal submission deadline was in March 2019 and the project is currently in its first stage with 56 proposals. The final selected algorithms may be used as lightweight alternatives to current AES-GCM and AES-CCM AEAD [Dwo07a, Dwo07b, NIS01] algorithms and SHA [NIS12, NIS15] hash function standards.

1.1 Most Lightweight PQC Algorithms Require a (Lightweight) XOF

In order to build real-world cryptosystems, one typically needs both symmetric and asymmetric cryptography. What is often overlooked is that many of the fastest PQC proposals also use symmetric cryptography *internally*.

In the case of asymmetric PQC algorithms, there is a large class of more lightweight NIST proposals based on (R/M)LWE (Learning With Errors [Reg09], with Ring [LPR10] and Module variants [LS15]) and the related (R)LWR (Learning With Rounding [BPR12], also optionally in a ring). A partial list of such candidates more suitable for embedded and other "lightweight" targets include Round5 [GMZB+19], SABER [DKRV19], Kyber [SAB+19], NewHope [PAA+19], Three Bears [Ham19], FrodoKEM [NAB+19], Dilithium [BAA+19], and qTESLA [BAA+19].

Almost universally SHAKE [NIS15] or cSHAKE [KjCP16] XOFs are required by these algorithms for tasks such as seed expansion (e.g. for creation of a unique, random $\mathbf{A}$) and for operations required for IND-CCA security, such as the Fujisaki-Okamoto transformation [FO99] and its variants. Similar seed expansion functionality can also be achieved with AES [NIS01], possibly combined with a hash function if the seed or other input is not already key-sized. The use of AES may be beneficial on high-end systems that have fast AES instructions available (e.g. Intel AES-NI). Embedded systems rarely have such instructions.

Some PQC algorithms spend a vast majority of their computational resources just performing iterations of the Keccak-f[1600] permutation used by SHA3/SHAKE. A FrodoKEM implementation had a more than six-fold overall speedup on Cortex-M4 when the AES or cSHAKE seed expander was replaced with `xoshiro128`, a lightweight non-cryptographic "PRNG" [BFM+18]. Furthermore [HOKG18] found cSHAKE to be the (by far) the largest module in a FrodoKEM FPGA implementation. Replacing SHAKE with the lightweight stream cipher Trivium [CP08] and other optimizations led to 16-fold speedup of FrodoKEM on FPGA (with the same implementation footprint) in [HMOR19].

The call for LWC proposals did not, unfortunately, include an extensible-output function (XOF) in its submission profiles, and only a few of the submissions directly support this functionality. However we designed the SNEIK family specifically for this purpose [Saa19b].

1.2 On Permutation-Based Cryptography

We are now coming to the end of a decade that saw the rise of permutation-based cryptography, now widely seen as a superior to the more traditional approach of based on using a distinct, specialized algorithm for each cryptographic subtask.

In the 1990s and 2000s most of the component algorithms required to build a higher-level cryptosystem or protocol were almost completely independent designs with only a few shared components; a block cipher or a stream cipher for confidentiality, a specialized MAC algorithm or HMAC [BCK96, KBC97] for integrity protection, a message digest for signatures, and asymmetric primitives which were purely based on a number-theoretic operations and required no specific symmetric cryptography (just a PRNG). Apart from the use of hash functions for HMAC these elements had essentially nothing in common. A security controller would implement all of them as separate, essentially independent modules.

Block cipher AEAD (Authenticated Encryption with Associated Data) modes such as CCM[Dwo07b] and GCM[Dwo07a] offered a partial solution since no longer was it necessary to process bulk data with both a block cipher and MAC/hash function circuitry.

The Keccak team has led the way to show that essentially all symmetric cryptographic functionality required to build a complete cryptosystem can be derived from a single permutation; not just hashes and XOFs (SHA3/SHAKE [NIS15]), but also PRNGs [BDPA10] and AEADs [BDPA11]. The sponge-based designs and security bounds have significantly evolved since originally proposed, with innovations such as increased rate [JLM14] and full-state keyed sponge and processing of associated data [GPT15, MRV15].

BLINKER [Saa14] was one of the first full protocol constructions based on a single Permutation; later proposals include Mike Hamburg's lightweight STROBE protocol framework [Ham17], David Wong's DISCO [Won19], and the Xoodyak suite from the Xoodoo/Keccak team [DHP$^+$18]. We see fully permutation-based cryptography as an ideal solution for the symmetric needs of embedded and other lightweight protocols. The usage of a single primitive for all of the required tasks significantly reduces the firmware and hardware implementation footprint. Hardware area is almost directly proportional to energy consumption.

2 Round5 + SNEIK 1.1 = R5Sneik

We designed SNEIK, a first-round NIST LWC candidate [Saa19b], to specifically to support Round5 [BBF$^+$19, GMZB$^+$19], a NIST 2nd round PQC candidate. We have measured the implementation metrics of the resulting "R5Sneik" algorithm hybrid on a Cortex M4 - based STM32F407 microcontroller and compared those to Round5 on the same target; there are significant performance and code size advantages.[3]

R5Sneik has exactly the same lattice and ring parameters as the original Round5. Round5 uses SHAKE [NIS15] and cSHAKE [KjCP16] primitives in KEM computation, and additionally AES-GCM [NIS01, Dwo07a] as a DEM for public key encryption. All of these symmetric components have been replaced with SNEIK 1.1 - based primitives in R5Sneik.

2.1 Round5 and its Designators

Round5 is a merger between author's first-round NIST proposal HILA5 and Round2, another first-round proposal from a team led by Philips Research. Round5 is based on Learning With Rounding (LWR) and its ring version (RLWR) [BPR12]. Round5 has an exceptionally large number of variants; we wanted to give engineers as much choice as possible.

[3] Round5 and R5Sneik are implemented in the r5embed project: https://github.com/r5embed/r5embed

Round5 and R5Sneik parameter designators take the form:

$$\texttt{R5}xx\_yzzz\_f\texttt{d}$$

where xx is either `ND`, indicating a ring variant, or `N1` indicating a more general lattice variant. Furthermore y indicates the NIST security level and is one of $\{1, 3, 5\}$. This is a security claim of equivalent or better resistance against quantum and classical attacks than AES with 128, 192, and 256 - bit keys, respectively. The zzz part is either `KEM` indicating IND-CPA - secure KEM functionality or `PKE` indicating IND-CCA - secure public key encryption. We note that it is relatively straightforward to strip the AES-GCM based DEM from the `PKE` variants and use them as an IND-CCA secure KEMs. The f parameter either 0 or 5 and indicates whether or not a forward error correction code XEf is used. The code can always correct at least f single-bit errors.

The final "`d`" indicates that this is the final parameter set that was submitted to NIST for 2nd round evaluation. The Round5 team circulated earlier versions during development phase. For example earlier Cortex M4 performance figures published in CARDIS '18 [SBGM+19] were based on earlier parameter set "`b`".

Since the proposed `N1` variants do not have error correction, while the ring-based `ND` variants optionally do, this leads to a total of $2 \times (1 + 2) \times 3 = 18$ variants. There are three additional special-purpose parameter sets, bringing the total number to 21. Our Cortex M4 implelementation supports all but one of the 21 parameter sets, `R5N1_3PKE_0smallCT`, which has very large (165kB) public keys – but very short ciphertext. The first three columns of Table 3 summarize the external parameters of Round5; private and public key size and ciphertext (expansion).

2.2 SNEIK 1.1

SNEIK [Saa19a] is a first-round candidate in the NIST Lightweight cryptography effort. Shortly after its publication it was updated to SNEIK 1.1 in response to a differential flaw discovered by Léo Perrin [Per19] and used in a message forgery attack by Khairallah [Kha19]. The rather trivial fix involved a single additional 1-bit rotation – and was actually also suggested by Perrin in [Per19]. The current version is designated SNEIK 1.1 [Saa19b].

The SNEIK family of cryptographic primitives is built around a 512-bit variable-round cryptographic permutation, $f512$. The permutation is an ARX construction [KN10] and was specifically designed for low-end microcontrollers. The three basic modes are:

- **SNEIKEN**: Authenticated Encryption with Associated Data (AEAD).
- **SNEIKHA**: Collision Resistant Hash Function.
- **SNEIGEN**: Entropy Distribution Function (EDF).

Only SNEIKEN and SNEIKHA are official parts of the NIST proposal (matching the profiles in the NIST call for algorithms), while SNEIGEN is included only as an "informational" appendix as it does not directly map to any of the traditional classes of cryptographic primitives. These algorithms are available at various security levels; see Table 1 for a summary.

SNEIK 1.1 parameters used in R5Sneik instantiations. The SNEIKEN authenticated encryption algorithm is instantiated for PKE variants at NIST security levels 1, 3, and 5 in a straightforward fashion with SNEIKEN128/192/256.

R5Sneik uses one of SNEIGEN128/192/256 for seed expansion and sampling, depending on the security level. The capacity of the generator state matches κ, the size of the seed.

Table 1: Basic parameters for the SNEIKEN family of algorithms. One can approximate their relative performance characteristics from $^r/_\rho$. Block size is always $b = 512$.

Name	Type	Security	Rounds	Rate
SNEIKEN128	AEAD	2^{128} (NIST1)	$\rho = 6$	$r = 384$
SNEIKEN192	AEAD	2^{192} (NIST3)	$\rho = 7$	$r = 320$
SNEIKEN256	AEAD	2^{256} (NIST5)	$\rho = 8$	$r = 256$
SNEIKHA256	Hash	2^{128} (collision)	$\rho = 8$	$r = 256$
SNEIKHA384	Hash	2^{192} (collision)	$\rho = 8$	$r = 128$
SNEIGEN128	EDF	*(no claims)*	$\rho = 3$	$r = 384$
SNEIGEN192	EDF	*(no claims)*	$\rho = 4$	$r = 320$
SNEIGEN256	EDF	*(no claims)*	$\rho = 5$	$r = 256$

SNEIKHA256/384 is used to replace SHAKE128/256 in Round5's Fujisaki-Okamoto KEM IND-CCA [FO99, HHK17] transform in R5Sneik; SNEIKHA256 at NIST level 1 and SNEIKHA384 at levels 3 and 5. Even though the collision resistance of SNEIKHA384 is only at 2^{192} level, to its 384-bit capacity its pre-image and distinguishability properties should satisfy the requirements of the transform at NIST Level 5, especially if we consider the limited number of queries available.

2.3 On Entropy Distribution Functions

SNEIGEN is a seed expander with limited cryptographic strength – it is not an authenticated encryption or hash function algorithm per se, and therefore not part of the main proposal. It is intended for cryptographic applications that need "random-like padding", "lightweight mixing" with well-understood entropy flow properties, or a deterministic PRNG source with good statistical qualities.

The main security requirement for such a lightweight mixing function is captured in the term "Entropy Distribution Function" (EDF); once seeded with $n \leq c$ truly random bits (n bits of entropy), any n-bit output should also have close to n bits of randomness (entropy) when observed without joint information. Here $c = b - r$ is the capacity of the variant.

SNEIGEN is **not** claimed to be collision resistant, but full collisions are unlikely for outputs that are much larger than the c-bit input seed. Given more than c bits of output, an attacker may be able to distinguish SNEIGEN from random, and may also be able to derive the secret state or even the input seed from it. However, a targeted cryptanalytic effort is required to achieve this. SNEIGEN output should not be directly exposed to an attacker in a way that leads to the compromise of the secret state (i.e. it should only be used internally).

Algebraic Interaction. Another key requirement is that the output of an EDF should not interact algebraically with the arithmetic operations of any higher-level cryptographic primitive that uses it. This means that, as an example, a completely linear EDF probably should not be used to distribute entropy between other linear components; there is a possibility that some of the entropy will algebraically cancel out or that the shared algebraic structure can somehow be used to attack the higher-level primitive.

We claim that the ARX structure of SNEIK does not interact with common rings, lattices, and other similar number-theoretical structures. However, this must be analyzed on a case-by-case basis.

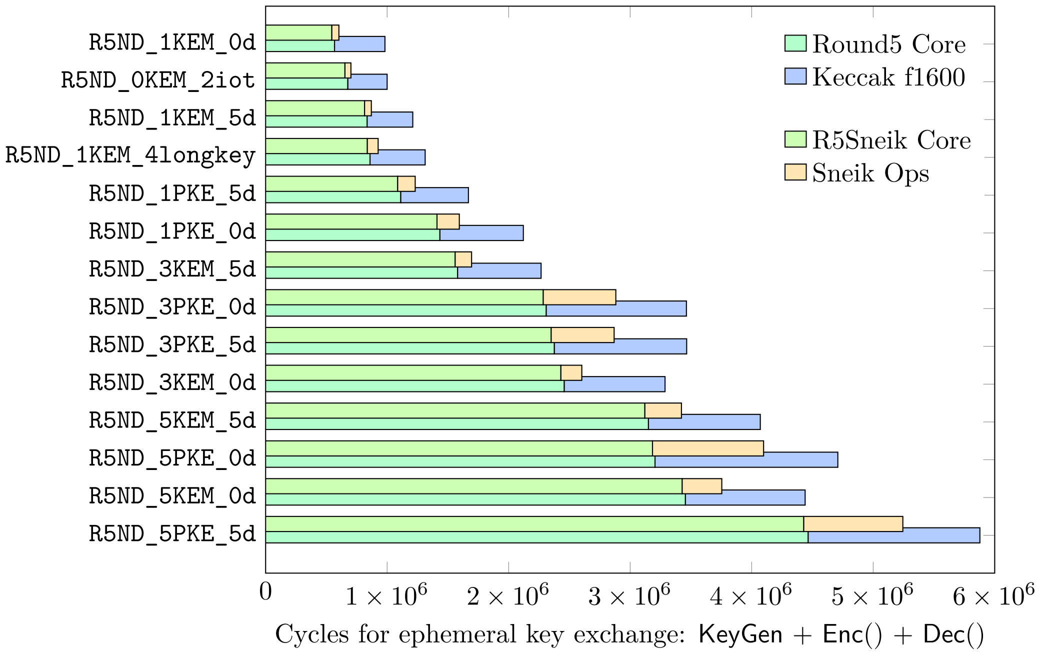

Figure 1: Comparison of the relative time Round5 and R5Sneik spend on XOF permutation computations alone; this can be up to 40% in case of `R5ND_1KEM_0d` with SHAKE. For IND-CCA (`PKE`) variants these profiles are just for the KEM part; DEM computation is excluded. The slight difference in the speed of green "core" asymmetric computations is caused by more complex padding operations and memory transfers required to handle the bigger Keccak-f[1600] permutation. Non-ring variants are not included due to scale.

The authors of [BFM$^+$18] argue that "good statistical properties" are sufficient for the public matrix **A** in a lightweight implementation of the FrodoKEM, another NIST PQC candidate. They use `xoshiro128**`, a very simple, fully XOR-linear PRNG (actually a seed expander) with a 128-bit state. We consider XOR-linearity a potential weakness as it may result in exploitable algebraic interaction with the lattice arithmetic.

2.4 Cortex M4 Implementations

We have ported almost all variants of Round5 to the ARM V7-m (Cortex M3/M4) platform. Most Cortex M3/M4 platforms are self-contained SoC microcontrollers with a clock frequency of 24-240Mhz, 1-1024kB of RAM, 2-4096kB of (Flash) program memory, and unit price between $1-$10. They are used as control logic in all kinds of electronic appliances.

We tested our implementation on the popular STM32F407 Discovery board[4]. This board was also used by the PQM4 [KRSS19] project in its evaluations. These results improve on our preliminary work reported in CARDIS 2018 [SBGM$^+$19]. Note that the NXP MK20DX256 chip used in those earlier experiments seems to use cycles somewhat (+10%) more efficiently than the ST STM32F407VGT6 chip used here.

Some key parts have now been implemented in ARM V7-m assembly language and use the UADD16 and USUB16 "SIMD" instructions. Further work on speedups is expected. Please

[4] STM32F4DISCOVERY: `https://www.st.com/en/evaluation-tools/stm32f4discovery.html`

refer to https://github.com/r5embed/r5embed for updated performance figures and to access the source code of the `r5embed` implementation.

Round5 and R5Sneik performance on Cortex M4. Table 2 gives preliminary cycle counts for the Cortex M4 implementation at submission time. Round5 maintains roughly equivalent or better performance than any comparable NIST candidate, while R5Sneik is up to 30 % faster than Round5 with the same lattice parameter set. Table 3 gives a summary of other resource utilization metrics of the implementation.

Permutation Implementations: Almost 90% size reduction. We used the Cortex M4 assembler-optimized Keccak-f[1600] implementation originally from the Keccak team. This is the same implementation as used by the PQM4 project [KRSS19]. It requires 12966 cycles to compute the 24 rounds of the 1600-bit permutation and has code size of 5260 bytes.

The SNEIK 1.1 permutation implementation used in testing requires 560 bytes of flash. Therefore all R5Sneik variants are consistently 5kB smaller than corresponding Round5 variants; this is significant if we look at ROM footprint in Table 3; Round5 with SHAKE is often twice the size of R5Sneik. This implementation of the SNEIK permutation requires about 203 cycles per round.

Figure 1 shows the relative portion that each one of the Ring (`R5ND_`) variants spends just computing the Keccak-f[1600] permutation. This can be up to 42.3% (`R5ND_1KEM_0d`). Permutation computation is a less dominant part of lattice versions (`R5N1_`): `R5N1_3KEM_0d` spends “only” 15.8% of its cycles computing Keccak permutations. The average across 20 measured variants is 28.4%. We note that the portion of execution time spent on SNEIK permutation by R5Sneik variants varies between 4.5% and 22.3%, with an average of 11.2%.

Note on timing attacks. Our prior work [SBGM+19] discussed a constant-time implementation that we had developed, but we are not including those numbers here. We note that the rejection sampler required for secret generation is “never” constant time but this rejection oracle is not really useful in attacks.

Cortex M4 does not have a data cache for its internal SRAM. The slower flash program memory has instruction cache (which can be disabled), and ST has added their own proprietary “ART Accelerator” for flash memory in STM32 [ST11]. Since Round5 and R5Sneik perform their operations in SRAM, timing attacks should not be an issue on this target.

It is also possible to use highly optimized “general” $\mathbf{Z}_{2^m}[x]$ multiplication methods such as those discussed in [KRS18] in the context of Round5. These techniques seem to require rather large code size, however.

3 Conclusions

We have demonstrated that by using an appropriate symmetric component (SNEIK 1.1) from the NIST LWC standardization effort to replace SHAKE in the Round5 post-quantum proposal we can relatively easily obtain a 30% direct performance boost and reduction of flash implementation footprint by up to 50% on Cortex M4 in some cases. The examined PQC/LWC variant, R5Sneik, is of course highly experimental at this stage.

We strongly encourage other PQC teams to explore integration with promising NIST LWC candidates in order to improve implementation characteristics on embedded (IoT) and hardware platforms.

Table 2: Side-by-side comparison of Round5 and R5Sneik performance on Cortex M4 (STM32F407VGT6 @ 24 MHz). All of the lattice parameters (and ring arithmetic code) are exactly the same. Numbers are in kilocycles (k = 1000 cycles) for **KG** = keypair generation, **Enc** = encryption or encapsulation, **Dec** = decryption or decapsulation, and **Tot** = full ephemeral key or message exchange. Note that the time required by the `PKE` variants to process authenticated messages with AES-GCM or SNEIKEN is included in these benchmarks.

	With SHAKE (Round5)				**With SNEIK (R5Sneik)**			
Algorithm	**KG**	**Enc**	**Dec**	**Tot**	**KG**	**Enc**	**Dec**	**Tot**
`R5ND_1KEM_5d`	391 k	573 k	244 k	1,209 k	280 k	419 k	164 k	864 k
`R5ND_3KEM_5d`	784 k	1,083 k	398 k	2,265 k	589 k	821 k	274 k	1,685 k
`R5ND_5KEM_5d`	1,427 k	1,951 k	692 k	4,071 k	1,212 k	1,658 k	536 k	3,407 k
`R5ND_1PKE_5d`	365 k	599 k	752 k	1,718 k	252 k	426 k	554 k	1,234 k
`R5ND_3PKE_5d`	785 k	1,212 k	1,514 k	3,512 k	590 k	1,013 k	1,259 k	2,864 k
`R5ND_5PKE_5d`	1,359 k	2,017 k	2,550 k	5,927 k	1,148 k	1,814 k	2,272 k	5,236 k
`R5ND_1KEM_0d`	357 k	470 k	153 k	982 k	231 k	292 k	75 k	599 k
`R5ND_3KEM_0d`	1,170 k	1,563 k	551 k	3,285 k	943 k	1,257 k	385 k	2,586 k
`R5ND_5KEM_0d`	1,588 k	2,134 k	716 k	4,439 k	1,358 k	1,826 k	553 k	3,738 k
`R5ND_1PKE_0d`	487 k	758 k	923 k	2,168 k	363 k	551 k	679 k	1,594 k
`R5ND_3PKE_0d`	798 k	1,233 k	1,482 k	3,513 k	603 k	1,037 k	1,241 k	2,881 k
`R5ND_5PKE_0d`	1,093 k	1,678 k	1,992 k	4,763 k	877 k	1,469 k	1,746 k	4,093 k
`R5N1_1KEM_0d`	6,522 k	4,328 k	1,100 k	11,952 k	5,862 k	3,356 k	517 k	9,735 k
`R5N1_3KEM_0d`	9,920 k	6,578 k	1,751 k	18,250 k	9,091 k	5,141 k	809 k	15,043 k
`R5N1_5KEM_0d`	34,751 k	19,845 k	4,290 k	58,886 k	33,183 k	17,469 k	2,406 k	53,058 k
`R5N1_1PKE_0d`	4,054 k	3,399 k	3,786 k	11,240 k	3,528 k	2,468 k	2,722 k	8,719 k
`R5N1_3PKE_0d`	17,011 k	11,160 k	12,834 k	41,007 k	15,788 k	10,202 k	11,163 k	37,154 k
`R5N1_5PKE_0d`	23,180 k	15,644 k	17,316 k	56,141 k	22,074 k	14,953 k	16,068 k	53,097 k
`R5ND_0KEM_2iot`	341 k	465 k	191 k	999 k	241 k	333 k	121 k	695 k
`R5ND_1KEM_4longkey`	419 k	624 k	268 k	1,313 k	291 k	445 k	183 k	920 k
`Saber` [DKRV19, KRSS19]	896 k	1,162 k	1,205 k	3,263 k	*Not Available*			
`Kyber768` [SAB+19, KRSS19]	977 k	1,147 k	1,095 k	3,218 k				
`ntruhps2048509` [CDH+19, KRSS19]	77,699 k	645 k	542 k	78,886 k				
`FrodoKEM640` [NAB+19, KRSS19]	81,902 k	86,306 k	86,447 k	254,654 k				

Table 3: ROUND5 and R5SNEIK Transfer, storage, implementation code size and stack usage requirements of various variants on an ARM7-m architecture system (such as Cortex-M3 or M4). **PK**, **SK**: byte sizes of the public key and secret key. **CT**: ciphertext size (KEM) or ciphertext expansion (PKE). The ciphertext expansion (for PKE variants) of R5Sneik is actually 8 bytes shorter due to a shorter message authentication tag. The **Code** size number excludes AES code used by the Round5 PKE DEM, but for R5Sneik the Sneiken DEM code is included. **KG**, **Enc**, and **Dec** give the memory (stack) usage for keypair generation, encryption / encapsulation, and decryption / decapsulation operations.

	Storage and Comms			**Round5 Footprint**				**R5Sneik Footprint**			
	Xfer	Priv	Xfer	ROM	RAM	RAM	RAM	ROM	RAM	RAM	RAM
Parameter Set	**PK**	**SK**	**CT**	**Code**	**KG**	**Enc**	**Dec**	**Code**	**KG**	**Enc**	**Dec**
`R5ND_1KEM_5d`	445	16	549	11,620	4,166	4,845	2,668	6,556	3,534	4,461	2,604
`R5ND_3KEM_5d`	780	24	859	12,926	5,854	6,949	3,716	7,866	5,278	6,653	3,716
`R5ND_5KEM_5d`	972	32	1,063	10,628	7,302	8,717	4,660	5,560	6,725	8,517	4,660
`R5ND_1PKE_5d`	461	493	652	12,460	4,262	5,701	5,700	7,334	3,622	5,309	5,308
`R5ND_3PKE_5d`	780	828	966	13,864	5,902	8,077	8,076	8,742	5,326	7,781	7,780
`R5ND_5PKE_5d`	978	1,042	1,317	11,650	7,342	10,357	10,356	6,512	6,766	10,093	10,052
`R5ND_1KEM_0d`	634	16	682	8,926	4,814	5,613	2,356	3,842	4,174	5,029	2,068
`R5ND_3KEM_0d`	909	24	981	9,046	6,326	7,877	4,716	3,986	5,750	7,573	4,676
`R5ND_5KEM_0d`	1,178	32	1,274	9,074	7,806	9,941	5,972	4,006	7,270	9,637	5,932
`R5ND_1PKE_0d`	676	708	772	9,872	4,814	6,733	6,732	4,738	4,182	6,285	6,284
`R5ND_3PKE_0d`	983	1,031	1,135	9,978	6,430	9,269	9,268	4,848	5,854	9,013	8,972
`R5ND_5PKE_0d`	1,349	1,413	1,541	9,968	8,366	12,325	12,324	4,842	7,790	12,061	12,028
`R5N1_1KEM_0d`	5,214	16	5,236	9,470	19,342	24,389	17,492	4,402	18,670	24,029	17,164
`R5N1_3KEM_0d`	8,834	24	8,866	9,518	26,686	35,477	27,372	4,442	26,086	35,141	27,116
`R5N1_5KEM_0d`	14,264	32	14,288	9,634	40,366	54,605	45,268	4,562	39,766	54,269	45,012
`R5N1_1PKE_0d`	5,740	5,772	5,820	10,128	19,894	31,381	31,388	4,994	19,222	31,021	31,028
`R5N1_3PKE_0d`	9,660	9,708	9,748	10,488	29,950	49,365	49,364	5,350	29,342	49,037	49,036
`R5N1_5PKE_0d`	14,636	14,700	14,740	10,348	37,046	66,501	66,500	5,004	36,438	66,165	66,164
`R5ND_0KEM_2iot`	342	16	394	10,076	3,494	4,005	2,084	3,340	2,854	3,485	1,884
`R5ND_1KEM_4longkey`	453	24	563	11,658	4,102	4,845	2,780	6,594	3,534	4,549	2,780

References

[AASA$^+$19] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, Daniel Smith-Tone, and Yi-Kai Liu. Status report on the first round of the nist post-quantum cryptography standardization process. Technical Report NISTIR 8240, National Institute of Standards and Technology, January 2019. doi:10.6028/NIST.IR.8240.

[BAA$^+$19] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qTESLA. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[BBF$^+$19] Hayo Baan, Sauvik Bhattacharya, Scott Fluhrer, Oscar Garcia-Morchon, Thijs Laarhoven, Ronald Rietman, Markku-Juhani O. Saarinen, Ludo Tolhuizen, and Zhenfei Zhang. Round5: Compact and fast post-quantum public-key encryption. In *PQCrypto 2019 – The Tenth International Conference on Post-Quantum Cryptography. Chongqing, China, May 8-10, 2019*, volume to appear of *Lecture Notes in Computer Science*. Springer, 2019. URL: https://eprint.iacr.org/2019/090.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996. doi:10.1007/3-540-68697-5_1.

[BDPA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-based pseudo-random number generators. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2010. doi:10.1007/978-3-642-15031-9_3.

[BDPA11] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011. doi:10.1007/978-3-642-28496-0_19.

[BFM$^+$18] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. Fly, you fool! Faster Frodo for the ARM Cortex-M4. IACR Cryptology ePrint Archive 2008/1116, November 2018. URL: https://eprint.iacr.org/2018/1116.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the*

Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012. doi:10.1007/978-3-642-29011-4_42.

[CDH+19] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. NTRU. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[CNS15] CNSS. Use of public standards for the secure sharing of information among national security systems. Committee on National Security Systems: CNSS Advisory Memorandum, Information Assurance 02-15, July 2015.

[CP08] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008. URL: https://doi.org/10.1007/978-3-540-68351-3, doi:10.1007/978-3-540-68351-3_18.

[DHP+18] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodoo cookbook. IACR Cryptology ePrint Archive 2018/767, November 2018. URL: https://eprint.iacr.org/2018/767.

[DKRV19] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. SABER. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[Dwo07a] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication SP 800-38D, November 2007. doi:10.6028/NIST.SP.800-38D.

[Dwo07b] Morris Dworkin. Recommendation for block cipher modes of operation: the CCM mode for authentication and confidentiality. NIST Special Publication SP 800-38C, May 2007. doi:10.6028/NIST.SP.800-38C.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999. doi:10.1007/3-540-48405-1_34.

[GMZB+19] Oscar Garcia-Morchon, Zhenfei Zhang, Sauvik Bhattacharya, Ronald Rietman, Ludo Tolhuizen, Jose-Luis Torre-Arce, Hayo Baan, Markku-Juhani O. Saarinen, Scott Fluhrer, Thijs Laarhoven, and Rachel Player. Round5. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[GPT15] Peter Gazi, Krzysztof Pietrzak, and Stefano Tessaro. The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO*

2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I, volume 9215 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2015. doi:10.1007/978-3-662-47989-6_18.

[Ham17] Mike Hamburg. The STROBE protocol framework. IACR Cryptology ePrint Archive 2017/003, January 2017. URL: http://eprint.iacr.org/2017/003.

[Ham19] Mike Hamburg. Three Bears. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017. doi:10.1007/978-3-319-70500-2_12.

[HMOR19] James Howe, Marco Martinoli, Elisabeth Oswald, and Francesco Regazzoni. Optimised lattice-based key encapsulation in hardware. Submitted for publication., May 2019.

[HOKG18] James Howe, Tobias Oder, Markus Krausz, and Tim Güneysu. Standard lattice-based key encapsulation on embedded devices. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):372–393, 2018. doi:10.13154/tches.v2018.i3.372-393.

[JLM14] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ security in sponge-based authenticated encryption modes. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 85–104. Springer, 2014. doi:10.1007/978-3-662-45611-8_5.

[KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. IETF RFC 2104, February 1997. doi:10.17487/RFC2104.

[Kha19] Mustafa Khairallah. Forgery attack on SNEIKEN. IACR Cryptology ePrint Archive 2019/408, April 2019. URL: https://eprint.iacr.org/2019/408.

[KjCP16] John Kelsey, Shu jen Chang, and Ray Perlner. SHA-3 derived functions: cSHAKE, KMAC, TupleHash, and ParallelHash. NIST Special Publication SP 800-185, December 2016. doi:10.6028/NIST.SP.800-185.

[KN10] Dmitry Khovratovich and Ivica Nikolic. Rotational cryptanalysis of ARX. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *Lecture Notes in Computer Science*, pages 333–346. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-13858-4_19, doi:10.1007/978-3-642-13858-4_19.

[KRS18] Matthias J. Kannwischer, Joost Rijneveld, and Peter Schwabe. Faster multiplication in $\mathbf{Z}_{2^m}[x]$ on Cortex-M4 to speed up NIST PQC candidates. IACR Cryptology ePrint Archive 2008/1018, October 2018. URL: https://eprint.iacr.org/2018/1018.

[KRSS19] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. PQM4: Post-quantum crypto library for the ARM Cortex-M4. (Accessed May 29, 2019.), 2019. URL: https://github.com/mupq/pqm4.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. URL: https://doi.org/10.1007/978-3-642-13190-5_1, doi:10.1007/978-3-642-13190-5_1.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015. Preliminary version was published as IACR Cryptology ePrint https://eprint.iacr.org/2012/090. doi:10.1007/s10623-014-9938-4.

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of full-state keyed sponge and duplex: Applications to authenticated encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 465–489. Springer, 2015. doi:10.1007/978-3-662-48800-3_19.

[NAB+19] Michael Naehrig, Erdem Alkim, Joppe Bos, Leo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[NIS01] NIST. Advanced Encryption Standard (AES). Federal Information Processing Standards Publication FIPS 197, November 2001. doi:10.6028/NIST.FIPS.197.

[NIS12] NIST. Secure Hash Standard (SHS). Federal Information Processing Standards Publication FIPS 180-D, August 2012. doi:10.6028/NIST.FIPS.180-4.

[NIS15] NIST. SHA-3 standard: Permutation-based hash and extendable-output functions. Federal Information Processing Standards Publication FIPS 202, August 2015. doi:10.6028/NIST.FIPS.202.

[PAA+19] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[Per19] Léo Perrin. Probability 1 iterated differential in the SNEIK permutation. IACR Cryptology ePrint Archive 2019/374, April 2019. URL: https://eprint.iacr.org/2019/374.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. Preliminary version appeared in STOC 2005. doi:10.1145/1568318.1568324.

[Saa14] Markku-Juhani O. Saarinen. Beyond modes: Building a secure record protocol from a cryptographic sponge permutation. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 270–285. Springer, 2014. doi:10.1007/978-3-319-04852-9_14.

[Saa19a] Markku-Juhani O. Saarinen. SNEIK. Round 1 submission to the NIST LWC Project, March 2019. URL: https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates.

[Saa19b] Markku-Juhani O. Saarinen. SNEIKEN and SNEIKHA v1.1: Authenticated encryption and cryptographic hashing. Technical report, PQShield Ltd., May 2019. URL: https://github.com/pqshield/sneik.

[SAB+19] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Round 2 submission to the NIST PQC Project, March 2019. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-2-Submissions.

[SBGM+19] Markku-Juhani O. Saarinen, Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Rietman, Ludo Tolhuizen, and Zhenfei Zhang. Shorter messages and faster post-quantum encryption with Round5 on Cortex M. In Begül Bilgin and Jean-Bernard Fischer, editors, *Smart Card Research and Advanced Applications. CARDIS 2018.*, volume 11389 of *Lecture Notes in Computer Science*, pages 95–110. Springer, 2019. URL: https://eprint.iacr.org/2018/723, doi:10.1007/978-3-030-15462-2_7.

[ST11] ST. How to achieve the lowest current consumption with STM32F2xx. Technical Report AN3430, STMicroelectronics, August 2011. URL: https://www.st.com/resource/en/application_note/dm00033348.pdf.

[Won19] David Wong. Disco: Modern session encryption. IACR Cryptology ePrint Archive 2019/180, February 2019. URL: https://eprint.iacr.org/2019/180.