From: D. J. Bernstein <djb@cr.yp.to>
Sent: Saturday, June 8, 2019 4:46 PM
To: pqc-comments
Cc: pqc-forum@list.nist.gov
Subject: [pqc-forum] ROUND 2 OFFICIAL COMMENT: SABER
Attachments: signature.asc

There are at least two problems with the first Saber theorem, Theorem 6.1. (Also with Round5 Theorem 2.6.1, which copied the Saber theorem and proof, modulo tweaks for the Round5 details; I won't bother filing this as a separate comment.)

1. Formally, the theorem statement is trivially correct and useless, since the statement fails to require an _efficient_ reduction. The most useful fix from the perspective of reviewers is to define the specific reductions before the theorem, and then use those definitions in the theorem statement.

2. More fundamentally, instead of claiming

  * security of Saber.PKE against all IND-CPA attacks under two
    Mod-LWR assumptions and a standard-model PRF assumption,

the theorem has to be weakened to claim merely

  * security of Saber.PKE against _ROM_ IND-CPA attacks under Mod-LWR
    assumptions.

In other words, cryptanalysts have to check not merely whether the hash function is a PRF, but whether the hash function as used inside the PKE enables any sort of non-ROM IND-CPA attacks.

Section 4.1 of my latticeproofs paper pinpoints the proof error. I'm not saying that I vouch for the correctness and applicability of the theorem with the above changes; I did only a sanity check, not a serious review.

---Dan

From: Sujoy Sinha Roy <Sujoy.SinhaRoy@esat.kuleuven.be>
Sent: Wednesday, September 18, 2019 4:28 PM
To: pqc-forum
Subject: [pqc-forum] Higher throughput Saber (OFFICIAL COMMENT)

Hi,

SaberX4 is a software implementation of Saber to achieve higher throughput, i.e., more KEM operations per second, on server machines with AVX2 support. A server needs to compute thousands of key exchanges every second; hence high throughput is desired in server applications.

The source code of SaberX4 is available at https://github.com/sujoyetc/SaberX4

SaberX4 batches four Saber KEM operations and computes them in parallel using AVX2 instructions, most of the time. With respect to the existing
AVX2 implementation of Saber, SaberX4 achieves nearly 38%, 45%, and 35% higher throughput for key generation, encapsulation, and decapsulation respectively.

The current implementation is a proof-of-concept. I believe there are scopes for further optimization to improve throughput. Some of the routines, such as message encoder/decoder, byte-string-to-polynomial and polynomial-to-byte-string conversions, binomial sampling, etc. are still executed serially, although they can be batched to achieve higher throughput.

Regards
Sujoy

From: pqc-forum@list.nist.gov on behalf of Sujoy Sinha Roy <Sujoy.SinhaRoy@esat.kuleuven.be>
Sent: Friday, March 27, 2020 7:02 AM
To: pqc-forum
Subject: [pqc-forum] Hardware implementation of SABER (OFFICIAL COMMENT)

Dear all,

We would like to let you know that our paper

"Compact domain-specific co-processor for accelerating module lattice-based key encapsulation mechanism" by Jose Maria Bermudo Mera and Furkan Turan and Angshuman Karmakar and Sujoy Sinha Roy and Ingrid Verbauwhede, (Accepted in DAC2020, https://eprint.iacr.org/2020/321.pdf

presents a co-processor architecture to speed up Saber. The HW/SW co-design runs more than 5 times faster than SW-only implementation.

The architecture was implemented in the Xilinx ZedBoard Zynq-7000 ARM/FPGA SoC Development Board following HW/SW codesign strategy. The paper shows how to implement the Toom-Cook polynomial multiplier in hardware.

Execution time (measured in million CPU cycles):

	SW only	SW/HW	Improvement
Key Generation	11.761	2.180	5.4
Encapsulation	14.944	2.762	5.4
Decapsulation	17.983	2.560	7.0

For a detailed description of the architecture, optimization techniques, results, and comparisons, please have a look at the paper at https://eprint.iacr.org/2020/321.pdf

More information regarding the Saber cryptosystem can be found on the fully updated website https://www.esat.kuleuven.be/cosic/pqcrypto/saber/

Regards
SABER Team

From: pqc-forum@list.nist.gov on behalf of Jose Mª Bermudo Mera <josebmera@gmail.com>
Sent: Wednesday, April 8, 2020 7:12 PM
To: pqc-forum
Subject: [pqc-forum] Updates on software implementations of Saber (OFFICIAL COMMENT)

Dear all,

We would like to let you know about the last updates on Saber:

1) Our latest paper "Time-memory trade-off in Toom-Cook multiplication: an application to module-lattice based cryptography" by Jose Maria Bermudo Mera and Angshuman Karmakar and Ingrid Verbauwhede, (Accepted in TCHES2020, https://eprint.iacr.org/2020/268.pdf)

shows how pre-computation and lazy interpolation can be applied to Toom-Cook multiplication to offer different time-memory trade-offs. The effect of such techniques is highly influenced by the underlying platform, but they can be combined for any SW implementation. Additionally, memory optimizations for embedded implementations are also shown.

The execution time for key generation/encapsulation/decapsulation is now:

Saber on AVX2 : 82 / 105 / 108 [x1000 clock cycles]
Saber on Cortex-M4 : 846 / 1098 / 1112 [x1000 clock cycles]

2) The Saber website (https://www.esat.kuleuven.be/cosic/pqcrypto/saber/) has been updated to include these last results, comparisons to other schemes and references to the last publications by our team or other teams.

3) Sources for software implementations of Saber on different platforms are now publicly available on github (https://github.com/KULeuven-COSIC/SABER). Further updates will come as we keep on improving Saber software and documentation.

Best regards,

SABER Team
--

From: pqc-forum@list.nist.gov on behalf of Andrea Basso <a.basso@cs.bham.ac.uk>
Sent: Wednesday, April 15, 2020 11:31 AM
To: pqc-forum
Subject: [pqc-forum] High-speed hardware implementation of Saber (OFFICIAL COMMENT)

Dear all,

We would like to announce our high-speed hardware implementation results for Saber.
Our instruction set coprocessor architecture for Saber implements all the building blocks (including CCA transformations) in the hardware.
The unified architecture is programmable and supports all KEM operations (key generation, encapsulation, and decapsulation).

During a KEM operation, the operand data is transferred to the coprocessor at once from a host processor, then all the computations are performed in the FPGA, and finally, the result is read by the host processor.

The implementation benefits from the modular algorithmic nature of Saber and its power-of-two moduli. For a security level similar to AES-192, the architecture achieves fast computation time and computes Saber key generation, encapsulation and decapsulation in only 21.8, 26.5, and 32.1 microseconds respectively and consumes only 9% of LUTs and 2% of flip-flops available in the XCZU9EG-2FFVB1156 FPGA.

We have uploaded a report to the ePrint. The report contains details of the architecture and comparisons with HW implementations of other schemes.
Hopefully, the report will be available soon.

Regards,
Andrea and Sujoy

--

From: Michiel Van Beirendonck <michiel.vanbeirendonck@esat.kuleuven.be>
Sent: Wednesday, June 17, 2020 8:44 AM
To: pqc-comments
Cc: pqc-forum
Subject: ROUND 2 OFFICIAL COMMENT: SABER

Dear pq-crypto enthusiasts,

We would like to share our latest results concerning masking Saber in software. We designed a first-order secure masked version of Saber on ARM Cortex-M4, and verified its security in our lab using the test vector leakage assessment methodology. The masked implementation of CCA-secure key decapsulation only incurs an overhead of 2.5x over the unmasked implementation. This can be compared to the reported overhead of 5.7x for masking NTT schemes such as NewHope and Kyber as reported by Oder et al. [1]. Our masked implementation takes just 2,833,348 cycles for the full decapsulation as can be seen in the following overview table:

Masking Scheme	**CPU cycles**		**Dynamic memory**	
	Unmasked	Masked	Unmasked	Masked
Our work	1,123,280	2,833,348 (2.52x)	6,320 bytes	11,656 bytes (1.84x)
CCA2-Secure and Masked RLWE [1]	4,416,918	25,334,493 (5.74x)	-	25,696 bytes

We note that 2,833,348 is a conservative number, which includes carefully constructed assembly to prevent microarchitectural leakage on the ARM.

Two properties of Saber make masking very efficient compared to other lattice-based schemes: the choice for power-of-two moduli and the usage of learning with rounding (LWR).

- Power-of-two moduli allow usage of efficient conversion algorithms between arithmetic and Boolean masking or between arithmetic maskings with different moduli, routines that are used frequently throughout the masked implementation.
- Learning with rounding eliminates the need for generating the error polynomials using the binomial sampler. As the masked binomial sampling, which includes a conversion from Boolean to arithmetic masking, is among the most expensive operations in a masked implementation, the suppression of these operations gives a big efficiency advantage to masking Saber. Furthermore, it reduces the amount of randomness needed for executing the algorithm. This advantage will only get stronger for higher-order masked implementations.

In our masked implementation, we take advantage of Saber's power-of-two moduli to directly construct a primitive that performs logical shifting on arithmetic shares, i.e. the masked substitute for rounding and message decoding. We integrate a recent masked binomial sampler, which we likewise instantiate with power-of-two mask conversion, as well as adapt for the specifications of Saber.

More detailed information is available in the paper that has been published on our website (https://www.esat.kuleuven.be/cosic/pqcrypto/saber/files/maskedSaber.pdf), and will shortly also be available on eprint.

Best regards,

The Saber team

[1] Oder, T., Schneider, T., Pöppelmann, T., & Güneysu, T. (2018). Practical CCA2-Secure and Masked Ring-LWE Implementation. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(1), 142-174. https://doi.org/10.13154/tches.v2018.i1.142-174