Xoodyak, an update

Joan Daemen[2], Seth Hoffert, Silvia Mella[1], Michaël Peeters[1],
Gilles Van Assche[1] and Ronny Van Keer[1]

[1] STMicroelectronics
[2] Radboud University

Date: September 18, 2020

In this document, we highlight some new results on our candidate algorithm Xoodyak since its submission in March 2019. In particular, we cover the following topics:

- ~~n~~ew third-party cryptanalysis;
- ~~i~~mproved differential and linear trail bounds;
- ~~n~~ew hardware implementations and performance results;
- ~~n~~ew protections against side-channel and fault attacks;
- ~~a~~ tweak proposal to speed up the processing of short messages.

This document answers the call of the NIST LWC team on August 18 to provide updates on the candidate algorithms.

1 New third-party cryptanalysis

Since the submission of Xoodyak in March 2019, the following papers have been published.

In [9], Song et al. mount cube attacks on a Xoodoo-based authenticated encryption scheme following the same mode as Ketje. The authors succeed on the initialization phase reduced to 6 rounds of Xoodoo (out of the nominal 12). Despite that Xoodyak does not use the same mode as Ketje, there is nevertheless significant similarity between their initializations. Furthermore, the authors discuss the effects of switching from 5-bit to 3-bit χ between Keccak-p and Xoodoo, and argue that the narrower χ contributes to an increased resistance against cube-attack-like analysis.

In [12], Zhou et al. apply a conditional cube attack to Xoodyak with the permutation reduced to 6 rounds. In the nonce-misuse setting, their attack could recover the 128-bit key in about 2^{44} operations with negligible memory costs.

In [7], Liu et al. show that a zero-sum distinguisher can be mounted on Xoodoo[12]. This is not surprising given the current knowledge on such distinguishers on Keccak-p, see, e.g., [1]. This distinguisher on the permutation does not contradict our claim of hermetic strategy as it does not extend to Xoodyak (or to a sponge-based function on top of Xoodoo[12] in general), see [4, Section 4.1].

2 Improved trail bounds

The design of the Xoodoo permutation comes with lower bounds on both differential and linear trails. They were obtained by exhaustively exploring the space of all trails up to a given weight, as described in our original publication on Xoodoo [3].

Since the submission of Xoodyak in March 2019, we have extended the trail analysis and improved the bounds, in particular for 4 and 5 rounds. Table 1 shows the currently known lower bounds. The details on the techniques used to improve the bounds can be found in [5, Section 4.3.1].

Table 1: The weight of the best differential and linear trails (or lower bounds) as a function of the number of rounds.

# rounds:	1	2	3	4	5	6	8	10	12
differential:	2	8	36	≥ 74	≥ 94	≥ 104	≥ 148	≥ 188	≥ 222
linear:	2	8	36	≥ 74	≥ 94	≥ 104	≥ 148	≥ 188	≥ 222

3 Hardware implementations and performance results

We implemented Xoodyak in hardware and performed performance analysis for ASIC and FPGA.

The implementation makes use of the *Development Package for Hardware Implementations Compliant with the Hardware API for Lightweight Cryptography*, v1.0.3 [11]. The supported external data bus width is 32 bits. The number of rounds R performed in a clock cycle is configurable at design time and can be set to 1, 2, 3, 4, 6, or 12, namely, to one of the divisors of the total number of rounds of the permutation.

The code is publicly available on GitHub as part of the Xoodyak repository [8] and was submitted to the FPGA and ASIC benchmarking projects [10, 6].

To compute the throughput of our implementation, note that the processing of a n-bit block of data takes $n/32 + 12/R + 3$ clock cycles, where $n = 352, 192, 128$ for associated data, plaintext/ciphertext and message to hash, respectively.

3.1 ASIC results

We synthesized the circuit in STMicroelectronics 40nm technology with Synopsys Design Compiler version Q-2019.12-sp1.

Table 2 reports the area cost in kilogate equivalent (kGE) for different values of R and different target frequencies. A dash means that the worst critical path does not respect the given timing constraint and thus area results are not meaningful.

The throughput of the most interesting R-frequency pairs is reported in Table 3. It appears that small values of R are usually better in terms of speed per area, unless a small frequency is required.

Note that we also developed an alternative version that supports only authenticated encryption and decryption (i.e., without hashing), and we noticed that the performance gain in terms of area is negligible (between 0.24 and 0.58 kGE).

3.2 FPGA results

We performed the synthesis using Vivado v2019.2 and a board of the Xilinx Artix-7 family (xc7a12tcsg325-3), which has 8,000 lookup tables (LUTs) and 16,000 flip-flops (FFs). The board is one of those suggested by the Athena FPGA benchmarking project.

Table 2: Area cost (in kGE) of the ASIC implementation of XOODYAK (including hashing).

Freq. (MHz)	R=1	R=2	R=3	R=4	R=6	R=12
100	8.20	10.66	13.12	15.58	20.25	49.85
200	8.20	10.69	13.12	15.59	30.65	-
300	8.26	10.72	13.39	25.03	-	-
400	8.43	10.90	22.74	-	-	-
500	8.78	14.57	-	-	-	-
600	9.09	-	-	-	-	-

Table 3: Throughput (Gbit/second) for associated data (AD), encryption and hashing area, for some area/speed trade-offs of the ASIC implementation.

Area (kGE)	Freq. (MHz)	R	AD	Enc.	Hash
8.20	100	1	1.35	0.91	0.67
8.20	200	1	2.71	1.83	1.35
8.26	300	1	4.06	2.74	2.02
8.43	400	1	5.42	3.66	2.69
8.78	500	1	6.77	4.57	3.37
9.09	600	1	8.12	5.49	4.04
10.66	100	2	1.76	1.28	0.98
10.69	200	2	3.52	2.56	1.97
10.72	300	2	5.28	3.84	2.95
10.90	400	2	7.04	5.12	3.94
13.12	100	3	1.96	1.48	1.16
13.12	200	3	3.91	2.95	2.33
13.39	300	3	5.87	4.43	3.49
14.57	500	2	8.80	6.40	4.92

Table 4 reports resource utilization in terms of LUTs for different values of R and target frequencies. The number of FFs required is 480 in all cases. Also in this case, the cost of the variant supporting only authenticated encryption and decryption does not differ significantly from the reported values.

The throughput of the most interesting R-frequency pairs is reported in Table 5. Like for the ASIC, it appears that small values of R are usually better in terms of speed per area, unless a small frequency is required.

Table 4: LUTs utilization of the FPGA implementation of Xoodyak (including hashing). The number of FFs used is 480 in all cases.

Freq. (MHz)	R=1	R=2	R=3	R=4	R=6	R=12
50	1417	1970	2672	3271	4313	10982*
100	1414	1970	2671	-	-	-
200	1399	-	-	-	-	-

*the number of LUTs required is beyond the available resources.

Table 5: Throughput (Gbit/second) for associated data (AD), encryption and hashing area, for some area/speed trade-offs of the FPGA implementation.

Area (LUTs)	Freq. (MHz)	R	AD	Enc.	Hash
1417	50	1	0.68	0.46	0.34
1414	100	1	1.35	0.91	0.67
1399	200	1	2.71	1.83	1.35
1970	50	2	0.88	0.64	0.49
1970	100	2	1.76	1.28	0.98
2672	50	3	0.98	0.74	0.58
2671	100	3	1.96	1.48	1.16

4 Protections against side-channel and fault attacks

Xoodyak was designed with protections against side channel attacks in mind. The submission document mentions various techniques to help prevent them, and masking the evaluation of the Xoodoo permutation is one of them. However, masking and faults can interact and result in statistically ineffective fault attacks (SIFA). Fortunately, there exist techniques to implement Xoodoo's χ operation in a way that prevents SIFA [2].

5 A tweak to speed up short messages

In this section, we present a tweak we will apply, should Xoodyak advance to the next round. The tweak will not be on Xoodyak itself, but on the subset that is submitted to the competition, with the purpose of speeding up the processing of very short messages.

Xoodyak is quite flexible and proposes a wider interface than what NIST required for the Lightweight Cryptography Standardization Process. In [4, Section 6], we map the functionality that NIST requires to sequences of calls to the Xoodyak interface. In particular, for authenticated encryption, we propose a sequence that absorbs the nonce in a dedicated call to Absorb(nonce). This is in fact one of three ways to handle the nonce, see [4, Section 3.2.2].

To speed up short messages, we will integrate the nonce with the key ID upon initialization. Assuming an empty key ID, the sequence to encrypt would then simplify to:

```
CYCLIST(K, nonce, ϵ)
ABSORB(A)
C  ENCRYPT(P)
T  SQUEEZE(t)
return (C, T)
```

The advantage is that the key and the nonce are processed in a single call to the XOODOO[12] permutation, instead of 2 calls in the current case. The processing of short messages therefore benefits from 12 rounds less to compute.

References

[1] C. Boura, A. Canteaut, and C. De Cannière, *Higher-order differential properties of Keccak and Luffa*, Fast Software Encryption, FSE 2011, Revised Selected Papers (Antoine Joux, ed.), Lecture Notes in Computer Science, vol. 6733, Springer, 2011, pp. 252–269.

[2] J. Daemen, C. Dobraunig, M. Eichlseder, H. Groß, F. Mendel, and R. Primas, *Protecting against statistical ineffective fault attacks*, IACR Trans. Cryptogr. Hardw. Embed. Syst. **2020** (2020), no. 3, 508–543.

[3] J. Daemen, S. Hoffert, G. Van Assche, and R. Van Keer, *The design of Xoodoo and Xoofff*, IACR Trans. Symmetric Cryptol. **2018** (2018), no. 4, 1–38.

[4] J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, and R. Van Keer, *Xoodyak, a lightweight cryptographic scheme*, Submission to NIST Lightweight Cryptography Standardization Process (round 2), March 2019, https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/Xoodyak-spec-round2.pdf.

[5] ———, *Xoodyak, a lightweight cryptographic scheme*, IACR Trans. Symmetric Cryptol. **2020** (2020), no. S1, 60–87.

[6] M. Khairallah, T. Peyrin, and A. Chattopadhyay, *Lightweight cryptography asic benchmarking*, March 2020, https://github.com/mustafam001/lwc-aead-rtl.

[7] F. Liu, T. Isobe, W. Meier, and Z. Yang, *Algebraic attacks on round-reduced Keccak/Xoodoo*, IACR Cryptol. ePrint Arch. **2020** (2020), 346.

[8] S. Mella, *Xoodyak hardware code*, August 2020, https://github.com/KeccakTeam/Xoodoo.

[9] L. Song and J. Guo, *Cube-attack-like cryptanalysis of round-reduced Keccak using MILP*, IACR Trans. Symmetric Cryptol. **2018** (2018), no. 3, 182–214.

[10] M. Tempelmeier, F. Farahmand, E. Homsirikamol, W. Diehl, J-P Kaps, and Kris Gaj, *A comprehensive framework for fair and efficient benchmarking of hardware implementations of lightweight cryptography*, June 2020, https://cryptography.gmu.edu/athena/index.php?id=LWC.

[11] ———, *Development package for hardware implementations compliant with the hardware api for lightweight cryptography, v1.0.3*, June 2020, https://github.com/GMUCERG/LWC.

[12] H. Zhou, Z. Li, X. Dong, K. Jia, and W. Meier, *Practical key-recovery attacks on round-reduced Ketje Jr, Xoodoo-AE and Xoodyak*, Comput. J. **63** (2020), no. 8, 1231–1246.