PHOTON-Beetle Authenticated Encryption and Hash Family — Updated on Software Implementations

Designers/Submitters:
Zhenzhen Bao - Nanyang Technological University, Singapore
Avik Chakraborti - NTT Secure Platform Laboratories, Japan
Nilanjan Datta - Indian Statistical Institute, Kolkata, India
Jian Guo - Nanyang Technological University, Singapore
Mridul Nandi - Indian Statistical Institute, Kolkata, India
Thomas Peyrin - Nanyang Technological University, Singapore
Kan Yasuda - NTT Secure Platform Laboratories, Japan and
zzbao@ntu.edu.sg, avikchkrbrti@gmail.com, nilanjan_isi_jrf@yahoo.com,
guojian@ntu.edu.sg, mridul.nandi@gmail.com, thomas.peyrin@ntu.edu.sg,
yasuda.kan@lab.ntt.co.jp

Abstract. PHOTON-Beetle is an authenticated encryption and hash family, that uses a sponge-based mode Beetle with the P_{256} (used for the hash function PHOTON) being the underlying permutation. This report updates software implementations of PHOTON-Beetle on 8-bit microcontrollers and that on general-purpose computers.

Keywords: PHOTON-Beetle · software · implementation

1 Updated on Software Implementations on 8-bit AVR

Members of PHOTON-Beetle have small code size (ROM) and low RAM requirement when being implemented in bit-sliced way on 8-bit AVR microcontrollers. To show the speed and the possible trade-off between memory and speed, we present performance of two sets of our implementations in Table 1, one is targeted at optimizing ROM (avr8_lowrom), and the other is targeted at improving speed (avr8_speed). The cores of the implementations are all written in assembly; the main authenticated encryption, decryption, and hash functions have C APIs (we extended our previous pure assembly implementations to be compliant with the SUPERCOP API of AEAD and hash). The implementations were compiled using AVR8/GNU C Compiler 5.4.0 in Atmel Studio 7.0. The specific targeted device is AVR ATmega328P. The code sizes, RAM usage, and cycles are also measured using components of Atmel Studio 7.0.

From Table 1, when targeting at optimizing ROM (avr8_lowrom), PHOTON-Beetle-AEAD can be implemented with code size less than 2200 bytes, and PHOTON-Beetle-Hash can be implemented with code size less than 1100 bytes. Supporting hashing on top of AEAD costs very limited additional resources (less than 300 bytes of ROM); Supporting full functionality (authenticated encryption, authenticated decryption, and hashing), all PHOTON-Beetle-Pairs requires less than 2500 bytes of ROM, less than 100 bytes of RAM. For the primary pair, PHOTON-Beetle-AEAD[128] runs (executing both authenticated encryption and decryption) at an average speed faster than 8200 cycles per byte; PHOTON-Beetle-Hash[32] runs at an average speed faster than 6600 cycles per byte.

Table 1: Performances of implementations on 8-bit AVR MCU

PHOTON-Beetle Pairs	Functionality	avr8_lowrom			avr8_speed		
		RAM	ROM	Speed	RAM	ROM	Speed
PHOTON-Beetle-AEAD[128]+ PHOTON-Beetle-Hash[32]	AEAD	86	2136	8128.03	86	4084	4835.35
	Hash	54	1034	6566.27	54	2982	3860.66
	AEAD+Hash	86	2416	-	86	4364	-
PHOTON-Beetle-AEAD[32] + PHOTON-Beetle-Hash[32]	AEAD	74	2134	19789.79	74	4082	11596.39
	Hash	54	1034	6566.27	54	2982	3860.66
	AEAD+Hash	86	2414	-	86	4362	-

- RAM is in bytes, and is measured excluding those used for storing test vectors (including plaintexts, associated data, master key, ciphertexts, tags, nonce, etc.). ROM is in bytes, and is measured excluding the codes for generating test vectors and looping of calling the functions.

- Speed in cycles per byte, and is measured by using the total cycles divided by the total bytes of data (length of associated data is **from 0 to 32 bytes**, length of plaintexts is **from 0 to 32 bytes**.) So, for AEAD, the total data length is 34848 bytes; For Hash, the total data length is 528. For AEAD, the total cycles includes that takes both by 'crypto_aead_encrypt' and 'crypto_aead_decrypt'. Thus, for AEAD, the speed is cycles per 'encrypting' and 'decrypting' one byte. This measurement is in line with that of https://lwc.las3.de/.

- We extended our previous pure assembly implementations to be compliant with the SUPERCOP API of AEAD and hash. Due to this change, the updated ROM and RAM requirements are larger than that reported in our previous submitted document.

When targeting at improving speed (avr8_speed), PHOTON-Beetle-AEAD can be implemented with code size less than 4100 bytes, and PHOTON-Beetle-Hash can be implemented with code size less than 3000 bytes. Supporting hashing on top of AEAD costs very limited additional resources (less than 300 bytes of ROM); Supporting full functionality (authenticated encryption, authenticated decryption, and hashing), all PHOTON-Beetle-Pairs requires less than 4100 bytes of ROM, less than 100 bytes of RAM. Specifically, for the primary pair, PHOTON-Beetle-AEAD[128] runs (executing both authenticated encryption and decryption) at an average speed faster than 4900 cycles per byte; PHOTON-Beetle-Hash[32] runs at an average speed faster than 3900 cycles per byte.

To see how the speeds vary with length of short messages, we present detailed speed for the primary pair in Table 2. Compared with the performance of implementations of AES-GCM in [SKP20], the speed is slower but acceptable, the ROM and RAM requirements are much less.

The updated implementations on 8-bit AVR are available via https://github.com/PHOTON-Beetle/Software.

Performance on Benchmarking Project. The platform established by Sebastian Renner, Enrico Pozzobon, and Jürgen Mottok (introduced in https://lwc.las3.de/), provides benchmarks of software implementations of AEAD of the second-round candidates. This platform also provided benchmarks of our submitted two sets of AVR implementations of PHOTON-Beetle. From the result about time and ROM on Arduino Uno R3 (MCU board based on the 8 bit ATmega328P MCU) presented in https://lwc.las3.de/table.php, the primary member PHOTON-Beetle-AEAD[128] have remarkable low ROM requirement[1]. Within a reasonable increase on the ROM (but is still relatively small), it can achieve moderate speed.

[1]In the presented result in https://lwc.las3.de/table.php, the ROM requirement includes that used to generate and check the test vectors. Thus, there is an obvious deviation between the ROM requirement presented in Table 1 and that presented in https://lwc.las3.de/table.php.

Table 2: Detailed speed of the primary pair of PHOTON-Beetle on AVR 8-bit MCU (length of AD = 16 bytes)

Algorithms	Func.	Package Length mlen [B]					ROM	RAM
		8	16	32	64	128		
PHOTON-Beetle-AEAD[128]	Enc	2476.38	1858.72	1652.56	1487.50	1377.38	4084	86
	Dec	2483.00	1863.41	1655.48	1489.14	1378.24		
	Enc+Dec	4959.38	3722.13	3308.04	2976.64	2755.63		
PHOTON-Beetle-Hash[32]	Hash	4880.13	2433.06	3568.50	4135.70	4419.27	2982	54

Denote the length of the message package by mlen, and the length of associated data by adlen that equals 16, the speeds of AEAD are measured by using cycles divided by (mlen+adlen).

2 Software Implementations on General-Purpose Computers

For general-purpose computers, table-based and bitslice-based implementations are two possible choices. We implemented PHOTON-Beetle in these two ways, and the implementations are provided via https://github.com/PHOTON-Beetle/Software.

The bitslice-based implementations achieve better performances on a personal computer, which are summarized in Table 3.

References

[SKP20] Yaroslav Sovyn, Volodymyr Khoma, and Michal Podpora. Comparison of three cpu-core families for iot applications in terms of security and performance of AES-GCM. *IEEE Internet Things J.*, 7(1):339–348, 2020.

Table 3: Speed of bitslice-based implementations of PHOTON-Beetle on PC

PHOTON-Beetle-AEAD[128] authenticated encryption (cycles/byte)								
mlen/adlen	0	16	32	64	128	256	512	1024
0	-	261.20	196.90	161.30	146.20	138.30	145.70	152.00
16	293.30	217.70	197.80	184.50	171.50	160.10	149.30	149.00
32	228.70	200.80	186.10	184.70	167.10	157.90	149.90	139.60
64	185.40	177.80	171.90	159.40	151.70	147.80	146.50	140.50
128	167.50	169.80	173.30	168.80	160.40	184.20	144.40	141.80
256	155.90	156.40	153.90	154.60	152.90	147.80	148.20	145.30
512	170.80	170.10	155.50	153.30	150.00	148.20	145.10	142.30
1024	150.70	151.10	151.60	163.40	162.90	159.50	156.00	149.20
PHOTON-Beetle-AEAD[32] authenticated encryption (cycles/byte)								
mlen/adlen	0	16	32	64	128	256	512	1024
0	-	733.60	735.70	839.90	577.60	573.40	564.90	541.60
16	685.70	613.90	588.70	567.40	553.40	550.70	561.20	545.80
32	617.00	602.60	595.00	589.10	569.80	556.50	543.00	542.80
64	620.00	590.10	569.90	568.70	572.70	563.20	563.30	558.00
128	566.00	568.40	644.50	559.60	566.90	555.50	548.90	575.00
256	560.50	555.90	560.20	557.10	564.60	572.50	554.90	548.80
512	561.20	573.70	564.40	562.00	553.80	557.80	569.10	549.30
1024	570.10	583.80	588.70	580.90	561.50	568.80	553.40	543.90
PHOTON-Beetle-Hash[32] hashing (cycles/byte)								
mlen	0	16	32	64	128	256	512	1024
	-	431.90	650.70	726.90	834.70	764.70	539.40	541.80

mlen: length of messages in Bytes, adlen: length of AD in Bytes.

The programs are compiled using GNU gcc 7.5.0. The processor is Intel(R) Core(TM) i7-6700 (Skylake). Hyper-threading and Turbo Boost are turned off during timing. The timing method used was that in http://github.com/BrianGladman/AES.