# Updates on WAGE

Mark Aagaard[1], Riham AlTawy[2], Guang Gong[1], Kalikinkar Mandal[3], Raghvendra Rohit[4], and Nusa Zidaric[1]

[1] Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Canada
[2] Department of Electrical and Computer Engineering, University of Victoria, Victoria, Canada
[3] Faculty of Computer Science, University of New Brunswick, Fredericton, Canada
[4] University of Rennes, CNRS, IRISA, France

**Abstract.** In this document, we report the activity progress of the WAGE authenticated cipher after the round 2 submission to NIST. The following activities were performed on WAGE: additional security analysis, proposing side-channel countermeasures and their evaluation in hardware and software, additional feature as pseudorandom bit generator, and some usecase studies for Internet of Things (IoT) protocol applications. We also propose a tweak to increase the throughput of WAGE.

## 1 Introduction

WAGE is a hardware-oriented lightweight authenticated cipher in the ongoing NIST lightweight cryptography standardization competition [5]. It is designed while keeping hardware efficiency in mind which is desirable for RFID and sensor applications. WAGE [2] is published as a formal publication at IACR Transactions on Symmetric Cryptology (ToSC), Volume 2020, Special Issue 1 [4], where we have provided further analysis of the security of WAGE against diffusion and differential attacks (See Section 2). In addition to a sponge-based pseudorandom bit generator (PRBG), we have shown in [4] how WAGE can be configured as a WG-PRBG with proven statistical properties. WAGE is analyzed against the correlation power attacks on the microcontroller platform, more specifically on ARM Cortex-M4F. To prevent against side-channel (e.g., DPA) attacks in both hardware and software, a higher-order masking scheme for WAGE is proposed in the $t$-SNI security model [8]. We evaluated the hardware performance of protected WAGE on two different technologies. The detailed results can be found in [8]. An Athena compliance API for WAGE is developed and submitted to the GMU group for performance evaluation. Finally, we have considered applications of WAGE in IEEE 802.11X [9] and CoAP [15] protocols for IoT in [14]. We also propose a tweak in the operating mode of WAGE to improve its throughput. The detailed security analysis of the tweaked variant will be available soon.

## 2 Security Analysis

### 2.1 Additional Security Analysis by the Designers

We briefly highlight some analysis from [4, Section 3].

**Diffusion behavior.** We model the diffusion behavior of WAGE to show its resistance against meet/miss-in-the middle distinguishers. Let $S^i_{j,k}$ denote the algebraic normal form (ANF) of the $k$-th bit of word $j$ after the $i$-th round. We say WAGE achieves full bit diffusion at $i$-th round if $S^i_{j,k}$ is a function of $S^0_{j,k}$, $\forall\ j = 0, 1, \ldots, 36$ and $\forall\ k = 0, 1, \ldots, 6$. Note that both WGP and SB sboxes have the full bit diffusion property but the multiplication by $\omega$ mixes only two bits at a time.

Since WAGE adopts an NLFSR based design, the word at position 0 is mixed at a slower rate than others. Thus, it is sufficient to find $i$ for which $S^i_{0,k}$ achieves full bit diffusion $\forall\ k = 0, 1, \ldots, 6$. Table 1 depicts such a behavior for word 0 while for other words the details are provided in [4].

From Table 1, we observe that WAGE achieves full bit diffusion in 28 rounds. Thus, WAGE (with 111 rounds) provides a huge security margin against meet/miss-in-the middle distinguishers as 56 (= 28+28) rounds guarantee full bit diffusion in both the forward and backward direction.

Table 1: Diffusion behavior of 0-th word of WAGE

| $S^i_{0,k}$ | Round $i$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $k$ | 1 | 4 | 7 | 11 | 15 | 19 | 23 | 27 | 28 |
| $0, \cdots, 6$ | 1 | 1 | 8 | 22 | 36 | 84 | 168 | 252 | 259 |

**Analysis on input and output differences.** We have analyzed the differential properties of WAGE permutation by restricting the input and output differences at rate positions only, i.e., differences are allowed at words 8, 9, 15, 16, 18, 27, 28, 34, 35 and 36. This case analyzes the resistance of WAGE authenticated encryption against differential and linear attacks in a sponge mode.

**Analysis on the forgery attacks against improper domain separation.** The choice of domain separators is crucial in resisting forgery attacks. WAGE-$\mathcal{AE}$-128 uses 2-bit domain separators `0x01` and `0x02` while processing associated data and message blocks, respectively. We showed that 2-bits are enough to distinguish all cases including empty, partial and complete associated data and/or plaintext/ciphertext blocks.

### 2.2 Side-channel Analysis

**Analysis by the designers [8].** We have analyzed WAGE against correlation power attacks in the microcontroller environment. We detected secret key leakage from power consumption in the first 12 (out of 111) rounds of the WAGE permutation and requires about 10,000 power traces to recover the 128-bit secret key. To resist such attacks, we proposed a masking scheme for WAGE (see Section 3.1).

**Third-party analysis by Bellizia et al. [6].** Recently, Bellizia et al. have analyzed the leakage-resistance of NIST LWC round 2 candidates by classifying them into modes and primitives. WAGE along with ACE, ASCON, SPIX and Spook has been shown to be CIML2 and CCAmL1 secure. Adopting the [6, Def. 1 & 2], CIML2 refers to the *"ciphertext integrity with misuse-resistance (i.e., no constraint on nonces) and leakage in encryption and decryption"*, while CCAmL1 means *"chosen ciphertext security with nonce misuse-resilience (i.e., fresh challenge nonce) and leakage in encryption only"*. For the proof details, the reader is referred to [6].

## 3 Performance Results

### 3.1 Side-channel Protection in Hardware and Software

**Side-channel implementation of WAGE in hardware.** We proposed a higher-order masking scheme of the WAGE authenticated encryption cipher in the strong non-interference (SNI) security model. We investigated different masking schemes for S-boxes by exploiting their internal structures and leveraging the state-of-the-art masking techniques. To practically demonstrate the effectiveness of masking, we performed the test vector leakage assessment on the 1-order masked WAGE. We evaluated the hardware performance of WAGE for 1, 2, and 3-order security and provided a comparison with other NIST LWC round 2 candidates. For instance, the 1-order masking scheme of WAGE consumed about 11,177 GE in ASIC on the ST Micro 65 nm technology. For the details, we refer the reader to [8].

**Side-channel implementation of WAGE in software.** The higher-order SNI-secure scheme of WAGE also provides side-channel resistance for software (e.g., microcontroller) implementations. In our proposed scheme, we used the randomized lookup table approach for 7-bit S-boxes to implement them in software. For the analysis, we first converted the C code to the ARM Cortex-M4F environment. We then performed the standard test vector leakage assessment (TVLA) to validate that our masked implementation is free of the first-order leakage. We observed that power traces do not detect leakage till 60,000 time points for the protected implementation while only 600 time points are sufficient to detect leakage in case of the unprotected implementation. For the details, the reader is referred to [8].

### 3.2 Software Performance

**Third-party software performance evaluation.** WAGE is included in Rhys Weatherley's AVR framework [13] to measure the performance of NIST LWC round 2 candidates on 8-bit and 32-bit microcontrollers where the timings are compared against the ChaChaPoly algorithm. For the details, the reader is referred to [13].

### 3.3 Hardware Performance

In 2019, a comprehensive analysis of parallel implementations of WAGE was presented in [3], and we included the implementation results for four ASIC technologies in the round 2 submission document [2]. For example, unparallelized WAGE has the area of 2900 GE and reaches a throughput of 518 Mbps using ST Micro 65 nm library. Throughput of 878 Mbps was acheived with the 8$\times$ parallel implementation, with area 12150 GE. In [3], we also present energy per bit, measured as the average value while performing cryptographic operations over 8192 bits of data at 10 MHz, which for unparallelized WAGE ST Micro 65 nm implementation yields 20.0 nJ.

Recently, protected WAGE implementations using two ASIC technologies are reported in [8]. As was already mentioned, the smallest implementation requires 11,177 GE using ST Micro 65 nm library. In [2], we also included implementation results for Xilinx Spartan-3 and Spartan-6, and for Intel/Altera Stratix IV. New preliminary LWC API compliant [11] results for WAGE on a Xilinx Spartan-6 are shown in Table 2.

## 4 Features

### 4.1 Psudorandom Bit Generator from WAGE

We presented an additional feature of WAGE as a pseudorandom bit generator (PRBG), which has appeared in [4]. We proposed two alternatives on how to configure WAGE and generate pseudorandom bits with minimal overheads: a) One construction is Sponge-PRBG, and b) Our second construction is based on the WG transformation, called WG-PRBG, by reusing certain circuitry of WAGE. The Sponge-PRBG construction from a permutation is based on the standard sponge construction, which was introduced in [7]. In such a construction, in order to generate a pseudorandom sequence of longer length, reseeding is required, meaning that after outputting a certain number of bits the generator needs to reseed to further produce output bits using the current internal state, requiring an external source for reseeding. However, it is hard to guarantee the randomness properties of the produced bits or sequences mathematically. In WG-PRBG, we can mathematically ensure certain randomness properties. One limitation is that WG-PRBG can generate a limited number of bits securely. However, it can be used for an arbitrarily length nonces and IVs. Note that, for WG-PRBG, the WAGE permutation functionality needs to be tweaked. The reader is referred to [4] for further details on both constructions and their security analyses.

### 4.2 Applications and Use-cases

WAGE can be implemented using 2540 GE at 940 MHz with a throughput of 536 Mbps and energy consumption of 39.2 nJ on ST Micro 90 nm [3], which is suitable for the RFID and sensor network applications.

**Application of WAGE in IEEE 802.11X and CoAP protocols.** We have considered applications of WAGE in IEEE 802.11X [9] and CoAP [15] protocols for IoT. We have shown in [14] how to configure WAGE as a key derivation function (`KDF`) and a message integrity check (`MIC`) generation function for the use in the IEEE 802.11X and CoAP handshake mutual authentication and key establishment protocols. We benchmark the performances of WAGE for `KDF` and `MIC` functionalities and handshaking and data protection protocols on microcontrollers as a major portion of IoT devices are equipped with microcontrollers. Our experimental results show that WAGE take about 2,903, 2,955, and 2,808 ms to complete the IEEE802.11X authentication protocol using ATmega128, MSP430F2370, and Cortex-M3, respectively. For the data protection protocol, WAGE achieves a throughput of 41, 34 and 53 Kbits/s on ATmega128, MSP430F2370, and Cortex-M3, respectively to encrypt and authenticate a plaintext of 1024 bits and an associated data of 128-bits. More details can be found in [14].

### 4.3 Comparison with AES-GCM

Using [10], we developed a hardware implementation of WAGE compliant with LWC API [11]. As the hardware implementation results for WAGE by the GMU Cryptographic Engineering Research Group are not yet available, we provide our own preliminary results. Upper half of the Table 2 shows our post Place-and-Route static timing results obtained for balanced design goal with Xilinx ISE Project Navigator 14.4 for the Xilinx Spartan-6 FPGA (XC6LX16-CS324).

Table 2: Comparison with AES-GCM

| | | $f$ | Area | TP | TPA | TPA2 | |
|---|---|---|---|---|---|---|---|
| Cipher | FPGA | [MHz] | [LUT] | [Mbps] | [Mbps/LUT] | [kbps/LUT$^2$] | Ref |
| AEC-GCM | Spartan-6 | 144.7 | 3350 | 1683.6 | 0.503 | 0.15 | [12] |
| WAGE | Spartan-6 | 124.1 | 1325 | 69.7 | 0.053 | 0.04 | |
| | ASIC library | [GHz] | [GE] | [Gbps] | [Mbps/GE] | [bps/GE$^2$] | |
| AEC-GCM | ST Micro 65 nm | 0.5 | 20580 | 5.82 | 0.28 | 13.7 | [1] |
| WAGE | ST Micro 65 nm | 0.56 | 7205 | 0.31 | 0.04 | 6.1 | |

The WAGE Spartan-6 results are compared to AES-GCM reported in [12]. This preliminary results show that WAGE requires only 40% of the area used by AES-GCM, and reaches approximately 85% of AES-GCM clock speed. The higher speed and block-size of AES-GCM result in a higher throughput (TP) and throughput per area metrics (TPA). We added the throughput per area squared metric (TPA2). Then we synthesized the GMU's AES-GCM code [1] for ST Micro 65 nm library; the logic synthesis results are shown in the lower half of Table 2. WAGE implementation requires only 35% of the area used by AES-GCM, and has a slightly higher frequency. The AES-GCM throughput and the derived metrics are still higher, but in comparison, WAGE is closer to AES-GCM for ST Micro 65 nm than for the Spartan-6 FPGA.

## 5 Proposed Tweaks

The initialization (resp. finalization) phase of WAGE comprises three (resp. two) calls for the WAGE permutation. Specifically, under the current recommended parameters, the initialization (resp. finalization) runs 333 (resp. 222) rounds of the underlying permutation. We propose to *change the number of rounds of the* WAGE *permutation in the initialization, associated data processing and finalization phases from 111 to 74*, if WAGE is selected as a finalist. Our choice limits the number of permutation calls in the initialization (resp. finalization) runs 222 (resp. 148) rounds, which offers sufficient security margin. Additionally, the total number of rounds is reduced by at least 185 which when processing short messages gives the throughput a good boost, e.g., when processing a 128-bit message, and 128-bit AD, throughput is improved by a factor 1.3. Note that this tweak does not require any change in hardware area (except an additional comparator and multiplexer on the round counter), software code size or RAM. Furthermore, based on our analysis it does not pose any threats to the security of WAGE. We now briefly explain our initial security analysis and full details will be available soon.

**Permutation behavior.** Note that an adversary can only obtain plaintext/ciphertexts and tag as outputs. For the plaintext/ciphertext phase, we are using the full 111 rounds of WAGE permutation, so its behavior is close to that of a random permutation. For the tag generation phase, since the keys are absorbed in state in the finalization phase, the WAGE permutation is in fact called for $74 \times 2 = 148$ rounds before outputting the tag.

**Security against differential/linear attacks.** For 74 rounds of WAGE permutation, the minimum number of active sboxes is 59 (Case I: no constraints on input and output differences) and 72 (Case II: input and output differences are restricted to rate positions only). The corresponding maximum expected differential characteristic probability (resp. maximum expected linear characteristic squared correlation) in $\log_2(\cdot)$ scale are -236 (-299.7) and -288 (-365.7) for Case I and Case II, respectively. See [4, Table 4, 5] for more details.

In a single-key setting, an adversary can only inject the differences in nonce (which will pass through $74 \times 3$ rounds), associated data (a 64-bit block is processed with 74 rounds) and plaintext/ciphertext (a 64-bit block is processed with 111 rounds). In all the cases, the differential probabilities/squared correlation are way higher than the available 64-bit degrees of freedom and the prescribed data limit of $2^{64}$ bits.

## References

[1] George mason university: Source code for aes gcm v1.0 (jun 2016). https://cryptography.gmu.edu/athena/index.php?id=CAESAR_source_codes.

[2] Mark D. Aagaard, Riham AlTawy, Guang Gong, Kalikinkar Mandal, Raghvendra Rohit, and Nusa Zidaric. Wage: An authenticated cipher, 2019. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/wage-spec-round2.pdf.

[3] Mark D. Aagaard, Marat Sattarov, and Nuša Zidarič. Hardware design and analysis of the ACE and WAGE ciphers. NIST LWC Workshop 2019. Also available at https://arxiv.org/abs/1909.12338.

[4] Riham AlTawy, Guang Gong, Kalikinkar Mandal, and Raghvendra Rohit. Wage: An authenticated encryption with a twist. *IACR Transactions on Symmetric Cryptology*, 2020(S1):132–159, Jun. 2020.

[5] Lawrence Bassham, Cagdas Calik, Donghoon Chang, Jinkeon Kang, Kerry McKay, and Meltem Sonmez Turan. Lightweight cryptography: Round 2 candidates, 2019. https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[6] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography a practical guide through the leakage-resistance jungle. In *Crypto*, 2020.

[7] 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*, pages 33–47, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[8] Yunsi Fei, Guang Gong, Cheng Gongye, Kalikinkar Mandal, Raghvendra Rohit, Tianhong Xu, Yunjie Yi, and Nusa Zidaric. Correlation power analysis and higher-order masking implementation of wage. pages 1–25, 2020.

[9] IEEE 802.11 Working Group et al. 802.11ax - ieee draft standard for information technology – telecommunications and information exchange between systems local and metropolitan area networks. *IEEE Std*, 2019.

[10] Ekawat Homsirikamol William Diehl Jens-Peter Kaps Michael Tempelmeier, Farnoud Farahmand and Kris Gaj. Development package for hardware implementations compliant with the hardware api for lightweight cryptography, v1.0.3;. https://cryptography.gmu.edu/athena/index.php?id=LWC.

[11] Ekawat Homsirikamol William Diehl Jens-Peter Kaps Michael Tempelmeier, Farnoud Farahmand and Kris Gaj. Implementer's guide to hardware implementations compliant with the hardware api for lightweight cryptography, v1.0.1 (nov 2019). https://cryptography.gmu.edu/athena/LWC/LWC_HW_Implementers_Guide.pdf.

[12] Behnaz Rezvani, Flora Coleman, Sachin Sachin, and William Diehl. Hardware implementations of nist lightweight cryptographic candidates: A first look. Cryptology ePrint Archive, Report 2019/824, 2019. https://eprint.iacr.org/2019/824.

[13] Rhys Weatherley. Lightweight cryptography primitives, 2020. https://rweather.github.io/lightweight-crypto/performance_avr.html.

[14] Yunjie Yi, Guang Gong, and Kalikinkar Mandal. Implementation of lightweight ciphers and their integration into entity authentication with ieee 802.11 physical layer transmission, 2020. In submission at IEEE Internet of Things Journal.

[15] K. Hartke C. Bormann Z. Shelby, K. Hartke. The constrained application protocol (coap). RFC 7252, 2014. https://tools.ietf.org/pdf/rfc7252.pdf.