Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 15(4): 212-216, Dec. 2017 **Regular paper**

High Performance Implementation of SGCM on High-End IoT Devices

Hwajeong Seo*, *Member, KIICE*

Department of IT Convergence Engineering, Hansung University, Seoul 02876, Korea

Abstract

In this paper, we introduce novel techniques to improve the high performance of AE functions on modern high-end IoT platforms (ARM–NEON), which support SIMD and cryptography instruction sets. For the Sophie Germain Counter Mode of operation (SGCM), counter modes of encryption and prime field multiplication are required. We chose the Montgomery multiplication for modular multiplication. We perform Montgomery multiplication in a parallel way by exploiting both the ARM and NEON instruction sets. Specifically, the NEON instruction performed 128-bit integer multiplication and the ARM instruction performed Montgomery reduction, simultaneously. This approach hides the latency for ARM in the NEON instruction set. For a high-speed counter mode of encryptions for both AE functions, we introduced two-level computations. When the tasks were large volume, we switched to the NEON instruction to execute the encryption operations. Otherwise, we performed the encryptions on the ARM module.

Index Terms: ARM–NEON, Authenticated encryption, SGCM

I. INTRODUCTION

The rapid development of embedded processors are enabling technologies for various Internet of Things (IoT) services including autonomous vehicle, surveillance systems and home automation. In order to utilize the IoT services, the massive data packets from deployed IoT devices are gathered and processed to extract the information. Since data contains sensitive and private information, secure wireless network communications is important for IoT devices.

The well-known cryptography protocol is to use block cipher based authenticated encryption (AE), which ensures data integrity, authentication and confidentiality. Among many AE technologies, the Galois/Counter Mode of operation (GCM) is a well-known algorithm in network security. GCM is a standard in NASA Suite B, IETF IPSec, IEEE 802.1 and TLS due to its efficient computation and high security. However, GCM is vulnerable toward a short authentication tag, introducing a high probability of successful forgery attacks in certain cases. The vulnerability raises questions about secure modes of operation. In order to resolve the security problem, a variant of GCM, namely, the Sophie Germain Counter Mode of operation (SGCM) is available [1]. The SGCM utilizes prime field (known as Sophie Germain prime) multiplication, instead of the binary field multiplication, which is secure against forgery attacks that GCM is vulnerable. Similarly, SGCM has the identical GCM computation structure, except for the GHASH function. Unlike GCM, 128-bit prime field multiplication is required to perform the GHASH function. However, recent works on IoT processors (ARM–NEON) only cover GCM

Received 23 June 2017, **Revised** 20 July 2017, **Accepted** 07 August 2017
***Corresponding Author** Hwajeong Seo (E-mail: hwajeong84@gmail.com, Tel: +82-2-760-8033)
Department of IT Convergence Engineering, Hansung University, 116, Samseongyo-ro 16-gil, Seongbuk-gu, Seoul 02876, Korea.

Open Access **http://doi.org/10.6109/jicce.2017.15.4.212** print ISSN: 2234-8255 online ISSN: 2234-8883

implementations. In this paper, we first suggest new SGCM results on ARM–NEON processors. We used Montgomery multiplication for prime field multiplication. For high performance, we used both ARM and NEON instruction sets in a parallel way. In particular, the NEON instruction computes integer multiplication while the ARM instruction computes Montgomery reduction. For counter modes of encryption, we introduced two-level encryption techniques. For small tasks, the ARM instruction performs the encryption. When the number of encryptions exceeds multiple times of 12, the NEON instruction computes 12 encryptions at once. With these optimization techniques, we achieved high-speed SGCM on ARM–NEON processors.

This paper is organized as follows. In Section II, we introduce the previous cryptography implementations on ARM–NEON. In Section III, we present implementation techniques in terms of prime field multiplication and block cipher encryption on the ARM–NEON platform. In Section IV, we evaluate the performance and compare the results with previous works. Finally, we conclude the paper in Section V.

II. RELATED WORKS

The first evaluation of cryptographic algorithms on the ARM–NEON architecture was reported by Bernstein and Schwabe [2] in CHES'12. The authors showed that the NEON instruction set supports high-security elliptic curve cryptography (Curve25519) at surprisingly high speeds. They also summarized the useful NEON instructions set and efficient vectorized register handling for high-speed cryptography, and presented the experimental results for the NaCl library on a Cortex A8 processor. In 2013, Camara et al. [3] employed the vmull.p8 instruction to describe a novel software multiplier for performing a 64-bit polynomial multiplication and a fast software binary field F_{2^m} multiplication on the ARM–NEON architecture. Their implementation results emphasized the advantage of NEON instruction for high-speed binary ECC and GCM.

In SAC'13, Bos et al. [4] presented a parallel approach to compute interleaved Montgomery multiplication, which is suitable for SIMD architectures including NEON, SSE, and AVX. Seo et al. [5] revisited Bos et al.'s work [4], and introduced the Cascade Operand Scanning (COS) method for multi-precision multiplication with the goal of reducing read-after-write (RAW) dependencies in the propagation of carries and the number of pipeline stalls. As a follow-up work, Seo et al. [6] proposed a novel Double Operand Scanning (DOS) method to speed-up multi-precision squaring with non-redundant representations on SIMD architecture and investigated RSA-1024 and RSA-2048 on an ARM Cortex A9 and A15 cores. Recently, Azarderakhsh et al. [7] presented efficient techniques of lattice-based cryptography and ring-LWE implementation on ARM–NEON architecture.

Besides public-key algorithms, cryptographic engineers also evaluated the impact of performance using symmetric block ciphers on the ARM–NEON architecture. In [8], the authors proposed a parallel implementation of block cipher LEA on ARM–NEON and achieved a speed-up of roughly 50% compared to the previous fastest implementation on ARM without NEON engine. As a follow up work, Seo et al. [8] proposed optimized LEA encryption by performing ARM and NEON instructions in a mixed processing way, which hides the latency of the ARM instruction in NEON overheads. In 2014, Saarinen and Brumley [9] presented the results of authenticated encryption algorithm such as WhirlBob and StriBob on the NEON platform. In CT-RSA'15, Gouvea and Lopez [10] used NEON instructions vmull to multiply two 64-bit binary polynomials and presented an optimized yet timing-resistant implementation of GCM over AES-128 on ARMv8.

In our work, we first present SGCM implementations on the ARM–NEON architecture. We present several optimizations of NEON instruction and an implemented mode of operations together with recent high-speed LEA implementations [8].

III. PROPOSED IMPLEMENTATIONS

Prime field multiplication is used to perform integer multiplication and modular reduction. For integer multiplication, we need to consider the representation between redundant and non-redundant on the SIMD architecture. The redundant representation avoids carry-propagation during the middle round by leaving empty bits in the word to keep carry bits. In the last round, we handle the carry bits to get the results in non-redundant representation. This is an efficient approach for the SIMD architecture since the platform does not support status flags (carry/borrow). For this reason, many ECC implementations have used redundant representation to achieve high performance [2]. In the SGCM operation, GHASH variables should be exclusive-ORed with ciphertext, which is only computable in non-redundant representation. However, the intermediate results are in redundant representation and this cannot be exclusive-ORed. Under redundant representation, the representation transitions between redundant and non-redundant representations are required to perform an exclusive-OR operation in every round, which degrades the performance significantly. Alternatively, we selected the non-redundant representation and COS multiplication method, which is the fastest implementation ever reported in non-redundant representation. For modular multiplication

for SGCM, we performed the multiplication and modular reduction operations. For efficient modular reduction, there are two popular methods available. The first approach is fast reduction. Since the modulus for SGCM is constant value, the reduction process can be hard-coded in off-line. However, it requires a number of addition and subtraction operations to handle the carry and borrow bits in constant execution timing.

The other modular reduction technique is Montgomery's algorithm, which was originally introduced in 1985 [11] and has been widely deployed in real-world applications (RSA). The algorithm replaces the division into the relatively cheap multiplication operation. Given the intermediate result of multiplication (A×B), the intermediate results (T) are multiplied by the inverse of modulus (M') and the results are reduced by (R) and stored into (Q). Afterwards, following equation ((T+Q×M)/R) is conducted. Finally, the calculation of the Montgomery multiplication may require a final subtraction of the modulus (M) to get a fully reduced result in the range of [0,M). In order to execute the Montgomery reduction, the target variables are converted into the Montgomery domain. The conversion is performed in the initial and last rounds by conducting the Montgomery multiplication. However, the GHASH should be bit-wise exclusive-ORed with ciphertext in a normal domain. In order to match the domain for both ciphertext and GHASH to the normal domain, we introduce a hybrid Montgomery reduction, which uses both Montgomery and normal domains for GHASH and ciphertext, respectively.

We set the GHASH constant in the Montgomery domain (H×R) and the authentication value in the normal domain (A). In GHASH multiplication, the GHASH constant and authentication value are multiplied to output the intermediate results (T). Afterwards, the intermediate result is reduced to an authentication value ($T \times R^{-1}$) through the Montgomery reduction step. Since the intermediate result is a Montgomery domain, the authentication value is placed into a normal domain and the value can be directly bit-wise exclusive-ORed with cipher-text in a normal domain.

This computation is optimized again through instruction set optimizations. We mix-used both ARM and NEON instructions to perform modular multiplication in a parallel way. Since the ARM and NEON are independent modules, two different routines are issued simultaneously. The NEON engine executes integer multiplication by using COS method while the ARM module performs the Montgomery reduction, which hides the latency for the ARM instructions in the NEON instructions. The mode of operation requires that GHASH function together with the block cipher encryption. In order to achieve high performance on the ARM–NEON, we selected an ARX (Addition, Rotation, eXclusive-OR) based block cipher, namely LEA. LEA was introduced in WISA'13 [12] and is efficiently computable over the ARM–NEON processor, because the vectorized instruction set supports 4 32-bit ARX operations in a single instruction, which performs the word-wise operations of LEA in an efficient way [8, 11, 13].

Table 1. Two-level encryption technique

input: plaintext (P), number of encryption (N), minimum number of parallel computation (M)
output: ciphertext (C)

```
1: Q ← N/M
2: R ← N mod M
3: i ← 0
4: while Q>i do
5:    C[M·i+M-1,…,M·i] ←ENC(P[M·i+M-1,…,M·i]) {NEON}
6:    i ← i +1
7: i ← 0
8: while R>i do
9: C[M·Q+i] ←ENC(P[M·Q+i]) {ARM}
10: i ← i +1
11: return C
```

In WISA'16, Seo et al. [13] further optimized the LEA implementations on the NEON processor. In our SGCM implementations, we adopted their implementations to fully utilize the SIMD capabilities. In order to achieve high performance, the block cipher encryption is performed in two different steps. When the number of encryptions are larger than the optimal number for NEON based encryption, we perform the encryption in NEON instruction. Afterwards, the remaining encryptions are performed in the ARM instruction because performing a small number of encryptions on ARM is more efficient than on NEON. The detailed descriptions are available in Table 1.

In step 1–2, the optimal number of encryptions for NEON (Q) and ARM (R) are calculated. Afterward, the encryptions are first performed by NEON instruction in step 4–7. Finally, the remaining encryptions are performed by ARM instruction in step 9–12.

IV. RESULT

A. Target Platform

For this study, we selected a high-end IoT architecture, namely ARMv7. Specifically, we executed the implementations on an ARM Cortex–A9 processor, which supports the full functionalities of the ARMv7 architecture and the NEON engine. ARMv7 and NEON support 32-bit and 64/128-bit wise registers, respectively.

Table 2. Results of binary/prime field multiplication in clock cycles on ARM Cortex–A9 platforms

Binary field multiplication (GCM) [3]	Prime field multiplication (SGCM) proposed
201.97	173.79

Table 3. Results of SGCM in cycles/byte on ARM Cortex–A9 platform (version 1: ARM only LEA, version 2: 2-level ARM/NEON)

Length	LEA-SGCM ver1	LEA-SGCM ver2
64 bytes	46.20	48.38
256 bytes	34.30	22.52
1024 bytes	31.80	15.08

The ARMv7 instruction set performs 32-bit SISD operations in a single cycle. On the other hand, the NEON engine provides vectorized instructions in 16 8-bit, 8 16-bit, 4 32-bit, and 2 64-bit, performing multiple data in a single instruction. The ARMv7 processor is widely used in high-end IoT processors including mini computers and mobile devices. The proposed implementations were working on all ARMv7 architectures including Cortex–A7, A8, A15 without modifications of source codes.

B. Evaluation

We tested our implementations on the ARM Cortex–A9 processor, which is equivalent to the target processors used in previous works [3]. For fair comparison, we only employed a single core and the optimization level was set to -O3. The detailed comparison of the results is drawn in Tables 2 and 3 [3].

For the prime field multiplication operation for SGCM, we achieved an execution time of 173.79 clock cycles on the Cortex–A9 processor, while Camara et al.'s binary field implementation [3] required 201.97 clock cycles. Between the binary and prime field multiplications, the prime field multiplication showed better performance than the binary field multiplication, because the prime field multiplication utilizes 2 32-bit integer multiplication while the binary field multiplication exploits 8 8-bit polynomial multiplication. The original Montgomery reduction for SGCM required about 210 clock cycles [5]. This was much slower than the proposed method, because the proposed method avoids a number of multiplications.

In terms of SGCM implementations, we evaluated various operand lengths (64, 256 and 1,024 bytes). For the short operand (64 bytes), two-level encryption showed lower performance than the traditional approach since two-level encryption has a routine to select the proper encryption modes between the ARM and NEON instruction set, which causes some overheads. For medium length (256 bytes), the two-level encryption begins to show better performance than that of the traditional approach since the NEON instruction performs the LEA encryption in an efficient way. For the long operand (1024 bytes), the 2-level encryption accelerated the performance significantly.

V. CONCLUSION

In this paper, we introduced efficient implementations of SGCM on 32-bit ARM–NEON processors. For SGCM, 128-bit Montgomery multiplication was performed in both ARM and NEON instruction sets, simultaneously. The parallel approach hides the small overheads in large overheads. Specifically, the NEON instruction computes the multiplication part and the ARM instruction computes the Montgomery reduction part, respectively. For message encryption, we introduced a two-level counter mode of operations for workload balancing. When the number of encryptions was smaller than the minimum number of NEON encryptions, the encryption operations were performed in the ARM instruction. On the other hand, when the number of encryptions was over the minimum number of NEON encryptions, the encryption operations were performed in the NEON instruction first. Afterwards, the remaining encryptions were performed in the ARM instruction. This approach increases the throughput by taking advantage of the NEON instruction set. By putting these optimization technologies together, we obtained compact SGCM authenticated encryption on a 32-bit ARM–NEON processor.

ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2017R1C1B5075742).

REFERENCES

[1] M. J. O. Saarinen, “SGCM: the Sophie Germain Counter Mode,” 2011 [Internet], Available: https://eprint.iacr.org/2011/326.pdf.

[2] D. J. Bernstein and P. Schwabe, “NEON crypto,” in *Cryptographic Hardware and Embedded Systems–CHES 2012*. Heidelberg: Springer, pp. 320–339, 2012.

[3] D. Camara, C. P. Gouvea, J. Lopez, and R. Dahab, “Fast software polynomial multiplication on ARM processors using the NEON engine,” in *CD-ARES 2013: Security Engineering and Intelligence Informatics*. Heidelberg: Springer, pp. 137–154, 2013.

[4] J. W. Bos, P. L. Montgomery, D. Shumow, and G. M. Zaverucha,

"Montgomery multiplication using vector instructions," in *Selected Areas in Cryptography–SAC 2013*. Heidelberg: Springer, pp. 471–489, 2013.

[5] H. Seo, Z. Liu, J. Groschadl, J. Choi, and H. Kim, "Montgomery modular multiplication on ARM-NEON revisited," in *Information Security and Cryptology–ICISC 2014*. Cham: Springer International Publishing, pp. 328–342, 2014.

[6] H. Seo, Z. Liu, J. Groschadl, and H. Kim, "Efficient arithmetic on ARM-NEON and its application for high-speed RSA implementation," *Security and Communication Networks*, vol. 9, no. 18, pp. 5401–5411, 2016.

[7] R. Azarderakhsh, Z. Liu, H. Seo, and H. Kim, "NEON PQCryto: fast and parallel ring-LWE encryption on ARM NEON architecture," 2015 [Internet], Available: https://eprint.iacr.org/2015/1081.pdf.

[8] H. Seo, Z. Liu, T. Park, H. Kim, Y. Lee, J. Choi, and H. Kim, "Parallel implementations of LEA," in *Information Security and Cryptology–ICISC 2013*. Cham: Springer International Publishing, pp. 256–274, 2013.

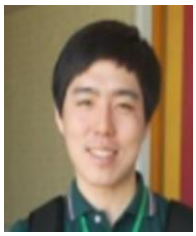
[9] M. J. O. Saarinen and B. B. Brumley, "Lighter, faster, and constant-time: WhirlBob, the Whirlpool variant of StriBob," 2014 [Internet], Available: https://eprint.iacr.org/2014/501/20141108:160918.

[10] C. P. Gouvea and J. Lopez, "Implementing GCM on ARMv8," in *Topics in Cryptology–CT-RSA 2015*. Heidelberg: Springer, pp. 167–180, 2015.

[11] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[12] D. Hong, J. K. Lee, D. C. Kim, D. Kwon, K. H. Ryu, and D. G. Lee, "LEA: a 128-bit block cipher for fast encryption on common processors," in *WISA 2013: Information Security Applications*. Cham: Springer International Publishing, pp. 3–27, 2013.

[13] H. Seo, T. Park, S. Heo, G. Seo, B. Bae, Z. Hu, L. Zhou, Y. Nogami, Y. Zhu, and H. Kim, "Parallel implementations of LEA, revisited," in *WISA 2016: Information Security Applications*. Cham: Springer International Publishing, pp. 318–330, 2016.



Hwajeong Seo
received the B.S.E.E. degree in 2010, and the M.S. degree in 2012 and the Ph.D. degree in 2016 in Pusan National University. He is currently an assistant professor in Hansung University.