# THE HARDWARE PERFORMANCE OF AUTHENTICATED ENCRYPTION MODES

Hristina Mihajloska[1]
UKIM, Faculty of Computer Science and Engineering
hristina.mihajloska@finki.ukim.mk

Abstract

Authenticated encryption has long been a vital operation in cryptography by its ability to provide confidentiality, integrity and authenticity at the same time. Its use has progressed in parallel with the worldwide use of Internet Protocol (IP), which has led to development of several new schemes as well as improved versions of existing ones. There have already been studies investigating software performance of various schemes. However, performance of authenticated encryption schemes on hardware has been left as an open question. We study the comprehensive evaluation of hardware performance of the most commonly used authenticated encryption modes CCM, GCM, OCB3 and EAX. These modes are block cipher based with additional authentication data (AAD). In order to make our evaluation fair, we have implemented each scheme with AES block cipher algorithm. In our evaluation, we targeted ASIC platforms and used 45 nm generic NANGATE Open Cell Library for syntheses. In each design, we have targeted minimizing the time-area product while maximizing the throughput. In the results, area, speed, time-area product, throughput, and power figures are presented for each scheme. Finally, we provide an unbiased discussion on the impact of the structure and complexity of each scheme on hardware implementation, together with recommendations on hardware-friendly authenticated encryption scheme design.

Key words: authenticated encryption, associated data, block ciphers, hardware perfomrance

## I. Introduction

When two parties want to communicate over network they should be sure that their communication will be secured. There are two main security goals: privacy and authentication. This means that one should never use encryption without also providing authentication on it.

Authentication encryption use has progressed in parallel with the worldwide use of Internet Protocol (IP), which has led to development of several new schemes as well as improved versions of existing ones [16]. The simplest solution for authenticated encryption is to use an encryption scheme for privacy and some hash function based message authentication code for authenticity. This scheme is known as Encrypt-then-MAC (EtM) composition of base schemes.

Over the past two decades, several strong ciphers and hash functions have been proposed [10], [1], [20], [7], [22], [14], [6], [3], [5] and even standardized. They can be used together in a EtM. While this provides guaranteed security, the same can not be said for its efficiency and performance. Using two completely different algorithms for encryption and authentication requires implementation of both algorithms separately. This not only means additional implementation effort for each algorithm, but it also means additional code space in a software and gates cost in a hardware implementation.

In order to overcome this barrier, combined schemes have also been proposed. Most of these schemes use a block cipher for encryption with additional invocations for authentication [24], [17], [2], [19]. There are also ciphers designed specifically for authenticated encryption [8], [11], [23]. More recently, authenticated encryption schemes are based on using permutation-based sponge functions [4] and attached a lot attention due to the SHA-3 competition where the winner was sponge hash function [21], Keccak [3]. Some of these existing schemes have been well-analyzed and recommended by NIST [24], [17]. Some have been standardized or become part of another standard [19]. However, a unified scheme or family of schemes is yet to be proposed. This open question and challenge are main goals to the recently-announced authenticated encryption competition CAESAR by NIST [9].

The rest of the paper is organized as follows. In Section II, we provide short descriptions of all investigated authenticated encryption modes. Hardware implementation specific details and results of all schemes are given in the following hardware evaluation section. Finally, we conclude our paper in Section IV with recommendations on hardware-efficient authenticated encryption scheme design.

## II. Authenticated Encryption Modes

We decided to investigate the authenticated encryption (AE) modes based on block ciphers. Out of the several existing schemes in the literature, we have picked up the most commonly used ones [24], [17], [2], [19], [13], some of which are even standardized or recommended by NIST. This scheme is very advantageous in terms of both hardware and software implementation. Instead of using two separate algorithms, only a single block cipher

[1] This work was partially financed by the Faculty of Computer Science and Engineering at the "Ss.Cyril and Methodius" University, Skopje, Macedonia.

is implemented with the accompanying mode wrapper and extra storage required for intermediate states for encryption and authentication. In this case only a single key is required just for the block cipher. Also encryption and authentication are performed during the one run of cipher function or first for encryption and then authentication. Most of the existing implementations use the NIST-standard AES [1] as the underlying cipher function. Some modes are even specified to be used only with AES. But most of them can also be used with another 128-bit cipher, while some of them are designed for any bit length ciphers as well. In our study, we have chosen CCM [24], GCM [17], OCB3 [13], and EAX [2] for investigation due to their widespread use in commercial applications. In the following subsections, each of these schemes will be described briefly.

- **CCM** is the abbreviation for *Counter with CBC-MAC*. CCM authenticated encryption with associated data essentially combines the counter (CTR) mode of encryption [15] with CBC-MAC authentication scheme [12]. To process each message block, a counter is encrypted with the underlying block cipher and the result is XORed to the message for ciphertext production. The message is also XORed with an *accumulator* which is then encrypted. The value of the accumulator corresponds to the internal message authentication state, and is kept being accumulated and updated until all the messages are processed. Its final state, the authentication tag after all blocks are processed. At the end of processing of each message block, the counter is also incremented for the next message block encryption. As a result of this operation sequence, each message block requires two encryption runs, resulting in low throughput. It can be improved by using two encryption module instances, which doubles the resource use. This is a primary trade-off in CCM implementations. On the positive side, decryption can also be realized using the same encryption scheme with ciphertext data instead of plaintext. Therefore, only the encryption functionality of the underlying cipher module is sufficient for CCM operation.
- **GCM** is the abbreviation for *Galois/Counter Mode*. It is very similar to CCM in operation. The encryption stage is identical, but authentication is realized via multiplication in $GF(2^{128})$ instead of the second encryption in CCM. The total number of encryptions required per message block is only one which results in a high throughput. The only penalty is the additional finite field multiplication, which can be effectively implemented by distributing the steps of multiplication to several cycles in parallel with the encryption operation. GCM mode is being deployed more and more in many applications due to its obvious advantages over CCM.
- **OCB3** is the abbreviation for a modified version of *Offset Codebook*. OCB3 also employs $GF(2^{128})$ as in the case of GCM, but in a simpler way. It does not require full multiplication, but only multiplication by powers of $z$ (the variable used in the polynomial representation of the finite field elements). To process the message block $i$, OCB3 performs a finite field multiplication of a nonce/key-dependent constant $L_0$ by the polynomial $z^j$, where $j$ is the number of trailing zeros in the binary representation of the block index $i$. The result (also known as tweak) is XORed to *accumulator* $a_1$. This accumulator is XORed to the message, encrypted, and then the result is XORed back with $a_1$ to generate the ciphertext. The message block is XORed to another accumulator $a_2$, which is used to generate the tag. A lookup table is used to hold precomputed values of $L_0 \cdot z^j$. Since the tweaks are computed prior to OCB3 operation and stored in the memory, the whole OCB3 operation is executed very effectively. However, the pre-computation phase must be executed with each new session key. OCB is usually referred to as the highest throughput mode. However, it also has drawbacks. Unlike CCM and GCM, it requires both encryption and decryption functionality in the underlying cipher. Furthermore, it is not a license-free scheme and can only be used subject to royalty fees expect certain applications.
- **EAX** authenticated encryption mode is a two-pass scheme: Encryption and authentication are performed separately. This makes EAX mode much slower than GCM or OCB3 modes, though it is *online* (which means it can process a stream of data using constant memory, without knowing the total data length in advance). EAX uses only the *encrypt* functionality of the block cipher, which makes it easy to fit into constrained implementations.

In this study we have implemented the target modes of authenticated encryption using AES block cipher.

- **AES** is the well-known and popular NIST-standard 128-bit block cipher. It is an iterated block cipher with three possible key lengths of 128, 192, and 256 bits and corresponding number of rounds of 10, 12, and 14, respectively. 128-bit key version is the most commonly used and benchmarked version, and we shall also be using it in our study. The 128-bit AES states is expressed as a $4 \times 4$ matrix of 8-bit (1 byte) words, and all the underlying substitution and permutation operations are applied on these bytes. Each of these bytes are considered as elements of the field $GF(2^8)$. One AES round consists of SubBytes, ShiftRows, MixColumns, and AddRoundKey operations. SubBytes is the substitution layer, where an 8-bit S-Box is applied to each byte of the state. ShiftRows is applied on the rows of the matrix state, and is defined as simultaneous left rotation of row $i$ of the state by $i$ positions. MixColumns multiplies each column of the state by a matrix defined over $GF(2^8)$ resulting in a new column. AddRoundKey adds (XORs) the state matrix with the round key. Each round key is derived from the secret key via the

key schedule process. The key schedule for decryption is the inverse of the key schedule encryption, and it requires another secret key. This secret key is the last round key of the encryption key schedule process. It can be stored as a second key or derived from encryption prior to each decryption, depending on the application requirements.

- **CCM-GCM-OCB3-EAX with AES:** These constructions use AES as the underlying cipher. Encryption and authentication of each message block requires two cipher module calls for CCM and EAX, and a single call for GCM and OCB3. In case of OCB3, decryption of the encrypted message block requires decryption functionality of the AES core. Therefore it requires execution of key expansion prior to each decryption session in order to generate the decryption key. It also requires an AES core which requires more resources in both software and hardware implementation.

## III. Hardware Implementation Results

In our hardware evaluation process, we have mainly targeted minimizing the time-area product for each scheme while trying to keep the throughput high. Before our evaluation, we have checked several comparison studies and observed that most of them suffer from minimal number of custom implementations. In other words, most authors prefer to implement their design(s) in only a few different configurations and using whatever technology library they have, and then they compare their very specific designs with other similarly specific designs. We have decided that such an approach would not be fair at all. Therefore we opted to implement every module ourselves using the same design approach.

Furthermore, we decided to use a publicly available standard-cell library to make our provide a common reference point accessible by everyone. We synthesized all our designs for 45 nm generic NANGATE [18] Open Cell Library using both Cadence Encounter RTL Compiler v10.1 and Synopsys Design Compiler vE-2010.12-SP2. The figures from both synthesis tools are close within a $\pm 7\%$ margin (in favor of either tool with no specific pattern). Since neither tool has an apparent advantage over the other, we only report the synthesis figures from Cadence. In all syntheses, typical operating conditions were assumed. All schemes are coded in Verilog and tested with the available test vectors. In the absence of test vectors, we generated them using Matlab prior to hardware implementation. The simulations were run using Modelsim v6.6c.

In this section, we present our results. As we have stated earlier, for each scheme (and module), our design target was the minimization of the time-area product, and the maximization of the throughput by avoiding any extra cycles. It might be argued that each scheme could be designed using much lower number of gates, or using flip-flops without enables, etc. Or it might also be argued that each design could be run faster by a pipelined and/or parallel design. However, we have specifically avoided design-specific optimizations in order to come up with a fair comparison. We also did not specifically targeted a low-power or low-area design. Instead, we provide results for the lightweight version of each scheme, where possible.

Table1: Hardware evaluation results

| | Area (kGE) | Speed (MHz) | Time · Area (ns · kGE) |
|---|---|---|---|
| CCM-AES | 18.9 | 247.89 | 76.24 |
| GCM-AES | 24.3 | 127.08 | 191.22 |
| OCB3-AES | 21.8 | 236.07 | 92.34 |
| EAX-AES | 17.6 | 127.52 | 138.02 |

Table2: Hardware evaluation results

| | Throughput (Mbps) | Power (mW) |
|---|---|---|
| CCM-AES | 1586.5 | 1.64 |
| GCM-AES | 1626.62 | 1.64 |
| OCB3-AES | 3021.7 | 2.68 |
| EAX-AES | 816.13 | 1.86 |

In the computation of throughput, we only considered the number of cycles spent for each message block. The initialization and finalization cycles and/or rounds were kept out of the calculation. They affect the average throughput with respect to the overall packet size. As the packet gets larger, these effects diminish.

In total, we have evaluated a total of 4 different schemes combined with AES block cipher. In the given tables, we provide the area, speed, time-area product, throughput and power comparisons for all evaluated schemes. In our comparisons, we favor none of the schemes. The selection of the corresponding scheme is left to designers for the target application. Likewise, any researcher can use the results while designing a new authenticated encryption scheme either for a specific application or to come up with a general scheme for the running competition.

## IV. Conclusion

In this paper, we have provided a comprehensive study on hardware implementations of authenticated encryption modes. We have only evaluated most popular block cipher based AE schemes. In each case, we used AES as block cipher to offer fair comparison. To the best of our knowledge, this is the first such comparison of different AE schemes. Considering the most commonly used AE ciphers, it will provide a reference for anyone who is willing to incorporate an existing AE scheme in a new design or anyone who is willing to come up with a new AE scheme targeting a specific application.

From a hardware designer's perspective, we have observed that the simpler any scheme, the better gate counts and lower delays it provides, thus resulting in a higher performance design. It should also be noted that

complex initialization (IV/counter generation) and finalization (tag generation) schemes should be avoided for hardware implementations (like in OCB3 and EAX).

At first it seems fair to conclude that it is very hard to come with a scheme favorable and suitable for both hardware and software. But then the AE scheme contradicts with this proposition. With so many existing AES designs and building blocks, it is a fairly simple task to come up with a very efficient hardware design for AE. In a similar way, thanks to the AES specific Intel instructions, it is also possible to write a very memory and speed effective code for this scheme.

In our opinion, the proposals to the authenticated encryption competition should cover all these angles, and incorporate building blocks that are very easily implementable both on software and hardware.

## Acknowledgment

The author wish to thank Ph.D Tolga Yalcin, Postdoc at Embedded Security Group from Ruhr University - Bochum, Germany for his valuable help regarding hardware implementation of the AE modes.

## References

[1] AES. *Advanced Encryption Standard*. FIPS PUB 197, Federal Information Processing Standards Publication, 2001.

[2] Mihir Bellare, Phillip Rogaway, and David Wagner. The EAX Mode of Operation. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *LNCS*, pages 389–407. Springer, 2004.

[3] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. *Keccak Specifications*, 2009.

[4] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the Sponge: Single-pass Authenticated Encryption and Other Applications. In *Proceedings of the 18th international conference on Selected Areas in Cryptography*, SAC'11, pages 320–337. Springer-Verlag, 2012.

[5] Martin Boesgaard, Mette Vesterager, and Erik Zenner. The Rabbit Stream Cipher. *eSTREAM Submission*, 2008.

[6] Andrey Bogdanov, Miroslav Knežević, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. SPONGENT: A Lightweight Hash Function. In *Proceedings of the 13th International Conference on Cryptographic Hardware and Embedded Systems*, CHES, pages 312–325. Springer-Verlag, 2011.

[7] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and Charlotte Vikkelsø. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[8] Andrey Bogdanov, Florian Mendel, Francesco Regazzoni, Vincent Rijmen, and Elmar Tischhauser. ALE: AES-Based Lightweight Authenticated Encryption. In *Fast Software Encryption - FSE 2012*, to appear.

[9] CAESAR. Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html .

[10] DES. *Data Encryption Standard*. FIPS PUB 46, Federal Information Processing Standards Publication, 1977.

[11] Daniel Engels, Markku-Juhani O. Saarinen, Peter Schweitzer, and Eric M. Smith. The Hummingbird-2 Lightweight Authenticated Encryption Algorithm. *IACR Cryptology ePrint Archive*, 2011:126, 2011.

[12] ISO/IEC 9797-1:2011. *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms Using a Block Cipher*.

[13] Ted Krovetz and Phillip Rogaway. *The OCB Authenticated-Encryption Algorithm*. Internet Engineering Task Force.

[14] Chae Lim and Tymur Korkishko. mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In *Information Security Applications*, volume 3786 of *LNCS*, pages 243–258. Springer, 2006.

[15] Helger Lipmaa, David Wagner, and Phillip Rogaway. *Comments to NIST Concerning AES Modes of Operation: CTR-Mode Encryption*, 2000.

[16] D. McGrew and K. Paterson. *Authenticated Encryption with AES-CBC and HMAC-SHA*. Internet Engineering Task Force.

[17] D. A. McGrew and J. Viega. The Galois/Counter Mode of Operation (GCM). *NIST Modes Operation Symmetric Key Block Ciphers*, 2005.

[18] NANGATE. 45 nm Open Cell Library.

[19] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A Block-cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, 2003.

[20] SHA. *Secure Hash Standard*. FIPS PUB 180-2, Federal Information Processing Standards Publication, 2002.

[21] SHA-3. Cryptographic Hash Algorithm Competition. NIST.

[22] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit Block Cipher CLEFIA (Extended Abstract). In *Fast Software Encryption - FSE 2007*, volume 4593 of *LNCS*, pages 181–195. Springer, 2007.

[23] Ruhma Tahir, Muhammad Y. Javed, and Ahmad R. Cheema. Rabbit-MAC: Lightweight Authenticated Encryption in Wireless Sensor Networks. In *International Conference on Information and Automation*, pages 573 –577, 2008.

[24] D. Whiting, R. Housley, and N. Ferguson. *Counter with CBC-MAC (CCM)*. Internet Engineering Task Force, 2003.