Hardware Performance of ELmD and ELmD(6,6)

Lilian Bossuet, Nilanjan Datta, Cuauhtemoc Mancillas-López, Mridul Nandi

Hardware Performance of ELmD and ELmD(6,6)

Lilian Bossuet *, Nilanjan Datta †, Cuauhtemoc Mancillas-López * and Mridul Nandi †
* Hubert Curien Laboratory UMR CNRS 5516 University of Lyon at Saint Etienne
† Indian Statistical Institute, Kolkata

***Abstract*—ELmD (Encrypt-Linear mix-Decrypt) is a block-cipher based efficient authenticated encryption scheme, which is nonce misuse resistant, fully pipeline implementable. ELmD is a candidate for CAESAR competition and it has been selected for the second round. In this document, we first consider two versions of ELmDv2.0 - (i) ELmD: full 10-round AES encryption-decryption, no intermediate tag, fixed tag size and (ii) ELmD(6,6): 6-round AES encryption-decryption, no intermediate tag, fixed tag size. We provide the full specification for both the version and present hardware implementation of the combined encryption-decryption for both these versions. We discuss and compare the hardware performance of these versions with other popular AE schemes like COPA and OCB.**

Keywords: Authenticated Encryption, Pipelining, Hardware Performance.

I. Introduction

The common application of cryptography is to implement a secure channel between two or more users and then exchanging information over that channel. Once the users have a shared key, either through the initial key set-up or key-exchange, they use this key to authenticate and encrypt the transmitted information using efficient symmetric-key algorithms such as message authentication code $Mac(\cdot)$ and (symmetric-key) encryption $Enc(\cdot)$. The encryption provides **privacy** or **confidentiality** (hiding the sensitive data M, we call it plaintext or message) resulting a ciphertext C, whereas a message authentication code provides **data-integrity** (authenticating the transmitted message M or the ciphertext C) resulting a tag T. An authenticated encryption or AE is an integrated scheme which provides both privacy of plaintext and authenticity or data integrity of message or ciphertext.

Now a days, cryptography community is putting a lot of effort of designing different authenticated encryptions. CAESAR[2], a competition for Authenticated Encryption is going on, in order to identify a portfolio of authenticated ciphers that offer advantages over AES-GCM and are suitable for widespread adoption. In the first round of CAESAR, 57 AE schemes were submitted and 29 of them have been qualified for the second round.

A. Hardware Implementation of Authenticated Encryption

In the literature there are many works about the hardware implementations of authenticated encryption based on block ciphers mainly standardized ones, GCM and CCM. As CCM has a sequential component for authentication since it uses CBC-MAC its implementation spectrum is not wide some examples of its implementation on FPGAs are [12] and [5]. GCM offers many ways for parallel implementations as is explored in [11] for FPGAs and [15] for ASIC.

For CAESAR submissions there are no much hardware implementations yet, some of the available implementations are ICEPOLE [13], SPRING [7] in [1] there are implementations results for Keyac, OCB, CLOC, PRIMATEs-GIBBON, PAEQ, PRIMATEsHANUMAN, POET, PRIMATEs-APE and AES-COPA. All the mentioned implementations are designed in sequential fashion, in this work we present pipelined implementations for ELmD and COPA. In general sequential implementations are small in terms of area but not too much as lightweight ones, also the throughput is better than lightweight but not enough for high speed applications. So we are trying to exploit the properties in this two modes to get a really fast implementation, able to process huge amount of data trying to keep the area reasonable.

B. Our Contribution

Here, we have considered ELmD, a CAESAR submission qualified for the 2nd round of the CAESAR competition. In this document, we first describe one of the 8 versions that have been submitted in CAESAR - ELmD with full 10 round AES encryption-decryption, no intermediate tag, fixed tag length (we call this version as ELmD) in section 2. Then in section 3, we describe a fast variant of this version - ELmD with 6 round AES encryption-decryption, no intermediate tag and fixed tag length. In section 5, we provide combined pipelined hardware implementation for both the versions and show that ELmD has good pipelined hardware performance. Moreover we compared the performance with COPA and make an important observation that ELmD requires only half of the area that COPA requires and yet offers higher throughput than COPA. We also make a comparative study of the hardware performance of ELmD with OCB-3, GCM, EME2 etc.

II. Specification of ELmD

ELmD is a block-cipher based authenticated encryption takes an associated data $D \in \{0,1\}^*$, a messages $M \in \{0,1\}^*$, a nonce N and generates a tagged-ciphertext $C \in \{0,1\}^{|M|+l_{tag}}$ in two steps, as described below. We compute $L = E_K(0)$ and use it to generate masks, during the tagged ciphertext generation. Here E_K is the block-cipher and instantiated by full 10-round AES.

A. Initial Value Generation.

Suppose we have a nonce N and an associated data $D = (D[1],\ D[2],\ \cdots,\ D[d-1],\ D[d])$. We set, $W'[0] = 0$

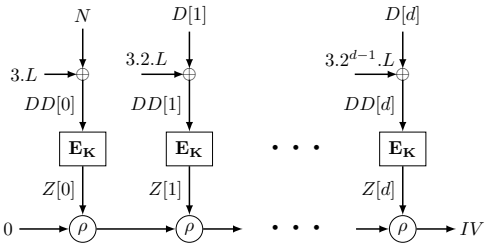

Fig. 1. Processing of Associated data in ELmD Authenticated Encryption : For complete final data block

and $D[0] = N$. Now, the processing of the $D[0]$, called the initial block and $D[..d]$ to generate the IV, is done as shown below.

$$\begin{aligned} DD[i] &= D[i] \oplus 3 \cdot 2^i \cdot L \quad \text{for } i = 0 \text{ to } d \\ Z[i] &= E_K(DD[i]) \quad \text{for } i = 0 \text{ to } d \\ (Y'[i], W'[i+1]) &= \rho(Z[i], W'[i]) \quad \text{for } i = 0 \text{ to } d \\ IV &= W'[d+1] \end{aligned}$$

INCORPORATING INCOMPLETE FINAL BLOCK. If the final associated data block ($D[d]$) is incomplete, then we make it complete by 10^* masking: $D[d] = D[d] \mid\mid 10^*$. The generation $DD[0], \cdots, DD[d-1]$ is identical as above but we modify generation of $DD[d]$ as follows (to distinguish from complete block associated data): $DD[d] = D[d] \oplus 3 \cdot 7 \cdot 2^{d-1} \cdot L$.

EMPTY ASSOCIATED DATA. If associated data is empty, we generate IV as follows: $IV = E_K(D[0] \oplus 3 \cdot L)$.

B. Tagged Ciphertext Generation.

The tagged ciphertext is generated using the message M and the IV, generated as described above using the nonce N and the associated data D. Suppose $M = (M[1], M[2], \cdots, M[l-1], M^*[l])$. If the final message block ($M^*[l]$) is incomplete, then we make it complete by 10^* masking: $M^*[l] = M^*[l] \mid\mid 10^*$.Now, the tagged ciphertext C is generated using the following equations :

$$\begin{aligned} W[0] &= IV \\ M[l] &= \oplus_{i=1}^{l-1} M[i] \oplus M^*[l] \\ M[l+1] &= M[l] \\ MM[i] &= M[i] \oplus 2^{i-1} \cdot L \quad \text{for } i = 1 \text{ to } (l+1) \\ X[i] &= E_K(MM[i]) \quad \text{for } i = 1 \text{ to } (l+1) \\ (Y[i], W[i]) &= \rho(X[i], W[i-1]) \quad \text{for } i = 1 \text{ to } (l+1) \\ CC[i] &= E_K^{-1}(Y[i]) \quad \text{for } i = 1 \text{ to } l \\ C[i] &= CC[i] \oplus 3^2 \cdot 2^{i-1} \cdot L \quad \text{for } i = 1 \text{ to } l \\ CC[l+1] &= E_K^{-1}(Y[l+1] \oplus 1) \\ C[l+1] &= CC[l+1] \oplus 3^2 \cdot 2^l \cdot L \end{aligned}$$

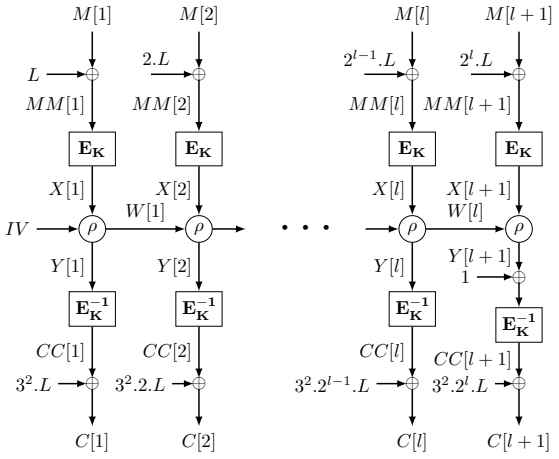

Fig. 2. ELmD Authenticated Encryption : For complete final message block

The algorithm returns tagged ciphertext $C = (\ C[1..l], (C[l+1])_{|M^*[l]|}\ )$.

INCORPORATING INCOMPLETE FINAL BLOCK. If the final message block ($M[l]$) is incomplete, then we make it complete by 10^* masking: $M[l] = M[l] \mid\mid 10^*$. The generation $MM[0], \cdots, MM[l-1]$ is identical as above but we modify generation of $MM[l]$ and $MM[l+1]$ as follows (to distinguish from complete block associated data): $MM[l] = M[l] \oplus 7 \cdot 2^{l-2} \cdot L$, $MM[l+1] = M[l+1] \oplus 7 \cdot 2^{l-1} \cdot L$.

III. ELMD(6,6): A FASTER VARIANT OF ELMD.

$ELmD(6,6)$ is a faster variant of ELmD where we instantiate 6 round AES encryption as block-cipher encryption E_K and 6 round AES decryption as block-cipher decryption E_K^{-1}.

From the structure of ELmD, we observe that, for some applications, even much lesser rounds of AES encryption or decryption is good enough to provide the desired security. Particularly we need to resist the collision in the upper layer encryption and want high randomness in the combined two layer encryption. Now, as 6 round AES gives good differential property (6-round AES is a good collision resistant hash) and the total of 12 $(= 6 + 6)$ rounds of AES in the combined upper-lower layer provides the desired randomness, we believe that choosing 6 rounds in both the layers will provide the desired security. Hence, we opt for the round parameter 6 as one of our recommended choice. The masking value L is defined as $L = E_K(E_K(0))$. The two block-cipher calls are used to ensure the required randomness. In this context, we also note that AES-6 has many key-recovery attacks but all those attacks uses the property that the chosen plaintexts has certain differential characteristic. Here these attacks are not applicable due to the upper layer masking and the randomness

of L.

To encrypt multiple messages under same associated data, one can have separate modules for AD processing and message-ciphertext processing. To ensure that AD processing module does not need any inverse, we define L to be $E_K(E_K(0))$ (instead of something like $E_K(E_K^{-1}(0)+const)$) when 6-round of AES is used.

IV. Hardware Implementation

In this section we describe the implementation of ELmD and compare it with COPA, EME2 (standardized for disk encryption [3]) which are algorithms with similar structure as ELmD and the standard for Authenticated Encryption with Associated Data GCM.

A. Design Decisions

Our implementation was realized to exploit the main features of ELmD. Below we list the design decisions:

1) **Pipeline Implementation:** ELmD is optimized for pipelined implementations, so we present a pipelined implementation of it. The size of data path is 128-bit same as block size of AES. Our implementation is suitable for bulk on-line authenticated encryption when the amount of data is huge. Two versions of ELmD were implemented one with ten rounds of AES in each layer denoted as $ELmD$ and other with only six rounds per layer denoted as $ELmD(6,6)$.
2) **AES-cores:** We use one AES-encryption core and one AES-decryption core. These cores were implemented in pipeline following the strategy in [8], it is an optimization of AES implementation for modern families of FPGAs with 6-inputs LUTs. Both encryption and decryption cores have ten pipeline stages, one per round as we are using 128-bit key. Only one key scheduling was implemented and it is shared between both cores. We add an additional pipeline stage for the multiplexer and necessary operations at the input of AES cores, in order to keep the frequency. So the initial latency for these cores is 11 clock cycles when the numbers of rounds is 10 and 7 clock cycles for 6 rounds.
3) **Multiplications:** ELmD needs many multiplications of a 128-bit string by a small constant such as $2, 3, 7$, we implement dedicated multipliers for each of them. This multipliers are really simples, just some shifts (implement as wires) and some $\oplus$'s. The field $GF(2^{128})$ is defined using the irreducible polynomial $P(x) = x^{128} + x^7 + x^2 + x + 1$.

B. Proposed Architecture

In the Figure 3 we show the proposed architecture for ELmD. The main components are AES-Enc and AES-Dec cores, they represent almost the 90% of the total area for this design. Register L store the value $L = E_K(0)$. All the components represented using rectangle with round corners contains dedicated multipliers, ones labeled as $x7$, $x3$ and $x3^2$ multiply their input by 7, by 3 and by 3^2 respectively. The two components $x2^i$ are used to generate masking values such as $L, 2L, 4L, ...$, internally they have a feedback to multiply each clock cycle the previous value by 2.

Block labeled with ρ computes the linear mixing function and its inverse. Mux4 provides the input to the block x^iL-1, this generates mask values, the input can be $3L$ for associated, L data for encryption and 3^2L for decryption. Using mux3 and mux4 AES-Enc receives the input, mux3 selects between associated data D_i and message blocks M_i to process them or checksum to compute the tag T. Mux1 selects input of AES-Enc between the masked value, masked value (mask multiplied by 7) when the block is incomplete or 0^{128} to compute L. Mux2 is in charge to provide the input to AES-dec, it selects the output of ρ xored with 1 in the last block cipher call which computes the tag, in others cases the input of AES-Dec is the output of ρ. Mux5 is used to feed the block x^iL-2, the input can be 3^2L for encryption and L for decryption.

Computation of the tag is in similar fashion as cipher text generation, just mux2 selects the $\oplus$ between the output of ρ and 1. For verification the tag T is fed after cipher text, then a checksum of the decrypted message is computed and compared with the output AES-dec.

Block chksum computes and stores the plaintext checksum while block checksumD do the same with the message decrypted. Verification is just a 128-bit comparator.

The Control Unit provides all the necessary control signals to the components of the architecture, it is a finite state machine that follows the corresponding algorithms for tagged encryption and verified decryption.

ELmD(6,6): Its implementation is very similar to the architecture for $ELmD$ the only different is that there is an additional input in Mux1 in order to compute $L = E_K(E_K(0))$, this additional input represents a feedback of AES-Enc core. In this case AES-Enc and AES-Dec cores contain only six rounds.

C. Timing analysis

Using the notation in *Specification of ELmD* we represent in the diagram of Figure 5 how the operations are computed in the time for our architecture. First of all L is computed as $L = E_K(0^n)$, this takes 11 clock cycles. Then the processing of the associated data starts (Z_i) and the block cipher encryption layer for message (X_i), using these values function ρ is computed and fed to the block cipher decryption layer and after 12 (11 from latency of the AES-dec core, 1 for the masking) clock cycles we can get a valid ciphertext in the output of the architecture. So the latency including the reset is computed as

$$Latency = 35 + d$$

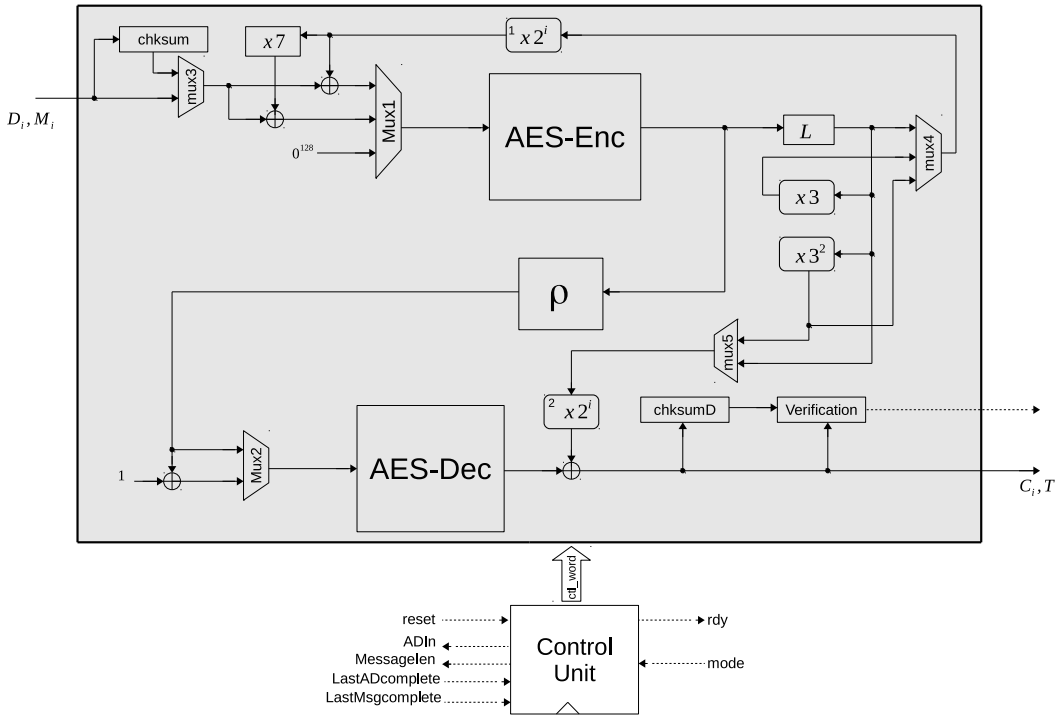

Fig. 3. Proposed architecture for ELmD.

where d is the number of $128-$bit block of associated data. Latency has to be understood as the number of clock cycles taken to output the first block of ciphertext.

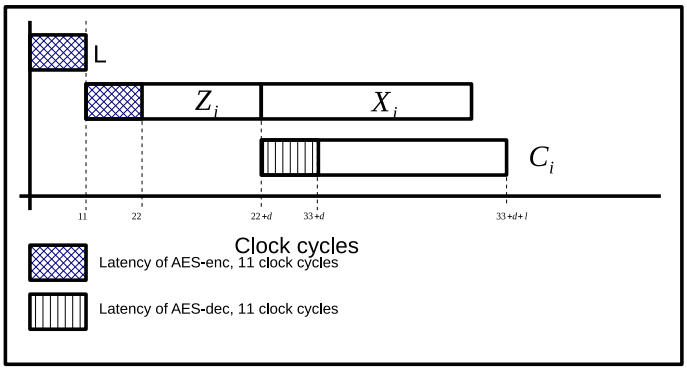

Fig. 4. Time diagram for $ELmD$. To get the real latency for C_i, two more clock cycles have to be added, one for the reset and other for final synchronization given a total time of $35 + d + l$ clock cycles. An additional clock cycle is used to give the tag T.

ELmD(6,6): The time diagram for it is shown in Fig. 5, in this case the latency for AES-enc and AES-dec is 7 clock cycles. The computation of L takes 14 because it implies two calls to AES-Enc. The latency to is $30+d$ taking into account the reset and synchronization of the output.

Before to analyze the results we give an overview of the implementation of COPA and OCB3.

Implementation of COPA: We implemented COPA to compare it with ELmD. We follow the same design decisions, we use two AES-enc cores and two AES-dec cores. The rest of the operations are like in ELmD mainly multiplications by small constants. The architecture for COPA is shown in Figure 6.

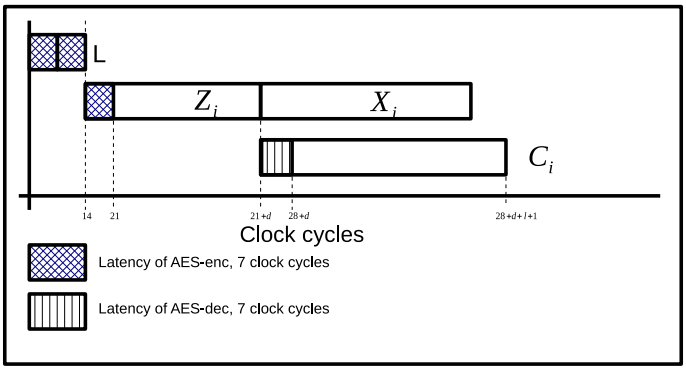

Fig. 5. Time diagram for $ELmD(6,6)$

Implementation of OCB: As for COPA we follow the same design decisions. In the case of OCB3 masking values are computed in a different way, it needs to do some precomputations of x^iL values and store them, hence OCB3 uses additional RAM memories (Block memories in FPGAs). Masking is generated using the number of trailing zeros so and special module was designed to compute them. OCB3 is optimized to save one block cipher call when a counter is used as nonce but it needs some clock cycles for the set up process, it is denoted as $Setup$, this process is execute only one time for some messages (if only if the nonce is a counter). The way to compute the initial value for masking is using a shift register, this function depends of the first six bits of the nonce, so the time taken by it is variable, we identify this function as $Stretch$. For more details of OCB consult [10].

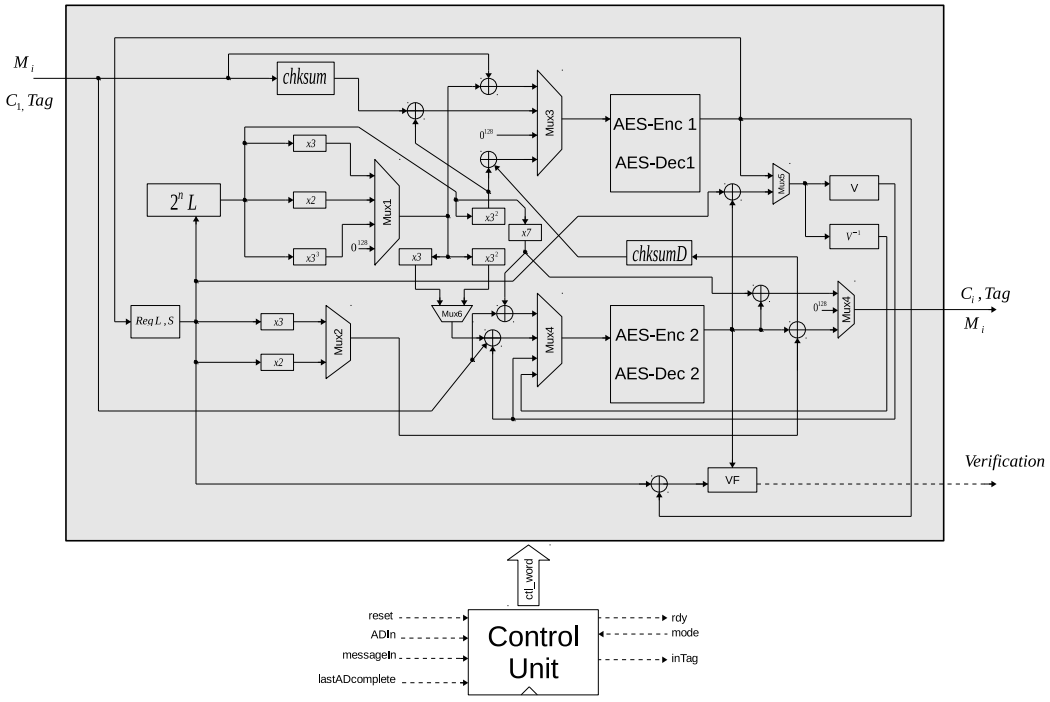

Fig. 6. Proposed architecture for COPA.

D. Results and Comparisons

In the Table I we show the results obtained after place and route using Xilinx ISE 13.4 and Virtex-6 (xc6vlx240t-2-ff1759) as a target device. Each Virtex-6 slice contains four 6-input-LUTs and eight flip flops (Virtex-5 slices used in [4] contain only four flip flops).

Our results show that the optimizations for area in ELmD design really save physical resources in comparison with COPA, basically the decision to use the second ECB layer always in decryption mode. Due to the fact that the changes between encryption and decryption algorithms of ELmD are really minimum and the block ciphers are always instantiated in the same mode, the first one is always encryption and the second is always decryption. For COPA the changes between encryption and decryption include also the block ciphers, when COPA is instantiated in decryption mode both block ciphers in the architecture have to be changed to decryption mode. This fact implies that while ELmD needs only one encryption core and one decryption core, COPA needs two of each one. So the logic resources used byt ELmD represents 50.28 % of the used by COPA.

Also the initial latency is less in ELmD, COPA needs 1.74 times more clock cycles to process the associated data in PMAC1 fashion since the computation of the last block needs to wait for all the encryptions of previous blocks.

The critical path for both designs is defined by AES-dec core which is significantly slower than AES-Enc core. About the throughput is almost the same for both since when the amount of data is huge the initial latency can be neglected, the throughput for ELmD is 30.03 Gbs while for COPA is 29.55 Gbs. So the real advantage of ELmD is that it offers a bit more throughput than COPA using half of the area.

In comparison with a similar implementation of GCM implemented in Virtex 5 FPGA, the difference in area between ELmD and GCM is almos 500 slices, GCM uses only one block cipher core and one 128-bit Karatsuba Ofman multiplier. In terms of area is much more expensive one AES-Dec core than a multiplier [8], this justify the difference in area. The speed for GCM is mainly defined by the critical path of AES-Enc core that's why GCM is faster than ELmD and COPA. Throughput is better for GCM but the security offers by ELmD and COPA is stronger than GCM security, ELmD and COPA can be used in nonce-misuse scenario as well where GCM is secure only in the nonce respecting mode. Finally the degradation in performance (not much) of ELmD and COPA in comparison with GCM can be compensated with the gains in security, since ELmD and COPA can be used in nonce-misuse scenario as well where GCM is secure only in the nonce respecting mode.

Comparing with EME2 [9] the design is more similar to COPA, they used the block cipher in the same mode (encryption or decryption) in two ECB layers, so the area for both are similar. The case of ELmD again the decision of used the two ECB layer in different mode, represent a good optimization against EME2. It is important to pointed out that EME2 needs intermediate storage, as large as the size of the input message since the masking between the two ECB layers depends of all the blocks in it. So EME2 can be implemented using only one AES-enc core and one AES-dec, since in EME2 both ECB layers are sequential in the sense that the second can be started only when the fist one has been finished. For ELmD and COPA it is possible to compute both ECB

TABLE I
PERFORMANCE OF ELMD AND COPA . AREA AND FREQUENCY FOR AES CORES ARE SHOWN IN THE BOTTOM PART. d IS THE NUMBERS OF 128-BIT BLOCKS OF ASSOCIATED DATA.

Mode	Area			Frequency (MHz)	Lantency clock cycles	Throughput Gbps
	Slices	LUTs	Flip Flops			
ELmD	5225	16967	5578	234.64	$35 + d$	30.03
COPA	10391	32845	8336	230.87	$61 + d$	29.55
AES-GCM [4] Virtex 5	4770	-	-	311	-	36.92
ELmD(6,6)	3150	10783	4018	238.68	$30 + d$	30.55
OCB3	5180	16879	5846	234.87	$11 + d + Setup + Stretch$	30.06
EME2[9]	10970	33350	9931	230.56	-	24.77
AES-10 pipelined encryption	2023	7301	2824	315.16	1	38.47
AES-10 pipelined decryption	2360	9020	2693	239.34	1	30.63
AES-6 pipelined encryption	1635	4523	2329	315.16	1	38.47
AES-6 pipelined decryption	1639	5353	2400	239.34	1	30.63

layers in a kind of pipeline using the output of the first as the input of the second, this is a characteristic of on-line modes of operation. We compared our implementations with EME2 only because the similarity of their structure but in fact the objective of EME2 is to provide a length-preserving strong pseudo random permutation suitable for disk encryption.

OCB3 uses almost the same amount of resources as ELmD, because they need one AES-enc and one AES-dec core. OCB3 require four block rams to store the precomputed values of $x^i L$, in this case we store 31 values. $ELmD(6,6)$ is much more small in terms of area due to the reduction in the number of AES rounds, so if security is comparable it would be a better option. Also is important to remember that OCB3 is secure only in the nonce respecting mode like GCM.

V. CONCLUSION AND FUTURE WORKS.

In this paper, we provide the hardware implementation of ELmD and ELmD(6,6) and compare it's performance with other existing constructions, with similar structure. As future work we can explore more range of parallelism for ELmD and try to achieve the requirements for the Ethernet standard IEEE 802.3ba, this specify a throughput of 100 Gbps [14]. This can be achieved using more AES-enc and AES-dec cores in parallel, each of them process different parts of the message.

REFERENCES

[1] Authenticated encryption fpga ranking. http://cryptography.gmu.edu/athenadb/fpga_auth_cipher/rankings_view.

[2] Caesar: Competition for authenticated encryption: Security,applicability, and robustness. http://competitions.cr.yp.to/caesar.html.

[3] IEEE Std 1619.2-2010: IEEE standard for wide-block encryption for shared storage media. IEEE Computer Society, March 2011. http://standards.ieee.org/findstds/standard/1619.2-2010.html.

[4] Karim M. Abdellatif, Roselyne Chotin-Avot, and Habib Mehrez. Fpga-based high performance AES-GCM using efficient karatsuba ofman algorithm. In Diana Goehringer, Marco Domenico Santambrogio, João M. P. Cardoso, and Koen Bertels, editors, *Reconfigurable Computing: Architectures, Tools, and Applications - 10th International Symposium, ARC 2014, Vilamoura, Portugal, April 14-16, 2014. Proceedings*, volume 8405 of *Lecture Notes in Computer Science*, pages 13–24. Springer, 2014.

[5] Ignacio Algredo-Badillo, Claudia Feregrino Uribe, René Cumplido, and Miguel Morales-Sandoval. Efficient hardware architecture for the AES-CCM protocol of the IEEE 802.11i standard. *Computers & Electrical Engineering*, 36(3):565–577, 2010.

[6] Lejla Batina and Matthew Robshaw, editors. *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*. Springer, 2014.

[7] Hai Brenner, Lubos Gaspar, Gaëtan Leurent, Alon Rosen, and François-Xavier Standaert. FPGA implementations of SPRING - and their countermeasures against side-channel attacks. In Batina and Robshaw [6], pages 414–432.

[8] Debrup Chakraborty, Cuauhtemoc Mancillas-López, Francisco Rodríguez-Henríquez, and Palash Sarkar. Efficient hardware implementations of BRW polynomials and tweakable enciphering schemes. *IEEE Trans. Computers*, 62(2):279–294, 2013.

[9] Debrup Chakraborty, Cuauhtemoc Mancillas-Lopez, and Palash Sarkar. Disk encryption: Do we need to preserve length? Cryptology ePrint Archive, Report 2015/594, 2015. http://eprint.iacr.org/.

[10] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.

[11] Stefan Lemsitzer, Johannes Wolkerstorfer, Norbert Felber, and Matthias Braendli. Multi-gigabit GCM-AES architecture optimized for fpgas. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 227–238. Springer, 2007.

[12] Emmanuel López-Trejo, Francisco Rodríguez-Henríquez, and Arturo Díaz-Pérez. An FPGA implementation of CCM mode using AES. In Dongho Won and Seungjoo Kim, editors, *Information Security and Cryptology - ICISC 2005, 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers*, volume 3935 of *Lecture Notes in Computer Science*, pages 322–334. Springer, 2005.

[13] Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICEPOLE: high-speed, hardware-oriented authenticated encryption. In Batina and Robshaw [6], pages 392–413.

[14] Michael Muehlberghuber, Christoph Keller, Norbert Felber, and Christian Pendl. 100 gbit/s authenticated encryption based on quantum key distribution. In Srinivas Katkoori, Matthew R. Guthaus, Ayse Kivilcim Coskun, Andreas Burg, and Ricardo Reis, editors, *20th IEEE/IFIP International Conference on VLSI and System-on-Chip, VLSI-SoC 2012, Santa Cruz, CA, USA, October 7-10, 2012*, pages 123–128. IEEE, 2012.

[15] Akashi Satoh, Takeshi Sugawara, and Takafumi Aoki. High-performance hardware architectures for galois counter mode. *IEEE Trans. Computers*, 58(7):917–930, 2009.