Maximizing the Throughput of Threshold-protected AES-GCM Implementations on FPGA

Jo Vliegen, Oscar Reparaz, Nele Mentens
imec-COSIC - KU Leuven
Kasteelpark Arenberg 10, 3001 Leuven, Belgium
Email: {jo.vliegen, oscar.reparaz, nele.mentens}@esat.kuleuven.be

***Abstract*—In this paper, we push the limits in maximizing the throughput of side-channel-protected AES-GCM implementations on an FPGA. We present a fully unrolled and pipelined architecture that uses a Boolean masking countermeasure (specifically, threshold implementation) for first-order DPA resistance. Using a high-end Virtex-7 device, we obtain a throughput of 15.24 Gbit/s. Since masked implementations require a stream of random bits for each execution, a high-throughput masked implementation requires a high-throughput pseudorandom number generator as well. This work determines how fast random numbers should be generated in order for ultra-high throughput, threshold-protected AES-GCM implementations to be feasible on FPGAs.**

I. Introduction

Real-time encryption and authentication are required in several high-end applications, like secure video streaming or videoconferencing. In practice, the term "real-time" is related to the latency and the throughput imposed by the application. When a very low latency and/or a very high throughput are required, the use of hardware implementation platforms is often unavoidable.

Regarding the choice of algorithms, a common way of providing both encryption and authentication is offered by the Galois Counter Mode (GCM), which is an authenticated mode of operation for block ciphers. GCM was designed by McGrew and Viega and is standardized as SP800-38D by NIST [1]. It uses established cryptographic primitives: the counter mode of operation (CTR) [2] for data confidentiality and a Carter–Wegman message authentication code [1] for data authentication.

AES-GCM is a well-understood, solid and popular authenticated encryption scheme. AES-GCM builds on the Advanced Encryption Standard (AES) [3] as underlying block cipher. The security AES-GCM provides has been rigorously proven [4]. AES-GCM can encrypt and authenticate data in a single pass, making AES-GCM suitable for low-latency and high-throughput applications. Here we choose AES-GCM as a case study for throughput optimization. There are other authenticated encryption schemes; notably, the ongoing CAESAR competition is currently evaluating proposals for authenticated ciphers [5].

Side-channel attacks are a cost-effective method to extract secrets, such as passwords or cryptographic keys, from embedded implementations [6]. A particularly effective strand of side-channel attacks is Differential Power Analysis (DPA) [7]. DPA is a technique that measures the instantaneous power consumption of an embedded device while executing the cryptographic implementation to extract the secret keys. DPA does not require expensive equipment and the attack scales economically; hence, it is customary to design implementations with built-in side-channel countermeasures to mitigate this threat.

A popular countermeasure for side-channel attacks is masking [8], [9]. Masking works by splitting secrets into several shares in a way that an attacker must recover the value of all shares to reconstruct the secret. In practice, masked implementations greatly increase the effort for an attacker to recover secrets, both in number of traces and computational effort. A masking scheme is normally designed for a specific algorithm. There are masking proposals tailored towards software [10], [11], [12] or hardware [13], [14]. In hardware, popular variants of Boolean masking are threshold implementations [15].

In this paper, we consider a threshold implementation of AES-GCM with three shares, aiming at first-order DPA protection. Both the AES block cipher and the authentication components of AES-GCM are fully masked. The Galois Field multiplier used in the authentication tag generation is Boolean masked. Inside the AES round, we follow the approach of Moradi et al. [16] to construct the shared S-box.

This is the first work that maximizes the throughput of threshold-protected AES-GCM implementations on an FPGA. It is therefore very useful for determining the speed requirements of FPGA-based pseudorandom number generators in ultra-high-speed, side-channel-secured applications of AES-GCM.

The paper is organized as follows. Section II contains background information on the AES-GCM scheme, the threshold countermeasure and the side-channel vulnerabilities of AES-GCM. Section III gives an overview of related work. The fully unrolled and pipelined architecture is described in Sect. IV. Section V discusses the results in terms of speed and occupied FPGA resources. Finally, Sect. VI draws conclusions and gives an outlook to future work.

II. Background

A. AES-GCM

Fig. 1 gives a conceptual visualization of GCM. As explained in Sect. I, GCM uses the CTR mode for data confidentiality. The counter is initialized with an initialization vector (IV) and is incremented providing $Y0$ to Yn. For every block of plaintext, the value of the counter is encrypted through the block cipher to generate a keystream. The XOR of the keystream with the plaintext generates the ciphertext.

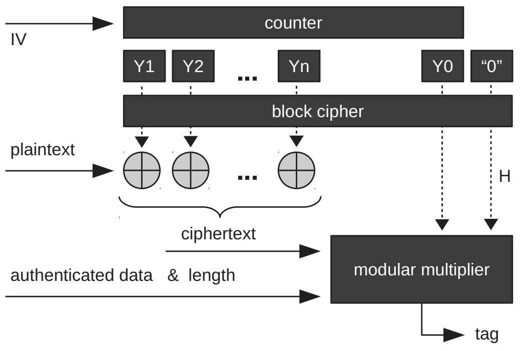
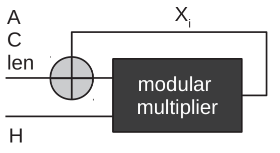


Fig. 1. Conceptual visualisation of GCM

For data authentication, the main operation is a modular multiplication. The first operand of this multiplication is the data authentication key H. The value H is constant and is obtained by encrypting an all-zeros input. The second operand of the multiplication is the XOR of the previous multiplication product with an input value that first comes from the authenticated data. These are data that are not encrypted but are protected by the MAC. After each block of authenticated data is processed, each block of ciphertext is XORed with the previous multiplication product and multiplied with H. Finally, the result is XORed with the previous multiplication product and multiplied with a 128-bit value, representing the length of the authenticated data and the plaintext. The resulting product of all modular multiplications is XOR-ed with the encrypted value of $Y0$, which generates the MAC/TAG. To increase the reader's understanding, Fig. 2 gives an overview of the consecutive inputs and outputs of the modular multiplier in GCM.



$$X_i = \begin{cases} 0 & \text{for: } i = 0 \\ (X_{i-1} \oplus A_i) *_M H & 1 \le i \le m-1 \\ (X_{m-1} \oplus A_m) *_M H & i = m \\ (X_{i-1} \oplus C_{i-m}) *_M H & m+1 \le i \le m+n-1 \\ (X_{m+n-1} \oplus C_n) *_M H & i = m+n \\ (X_{m+n} \oplus len) *_M H & i = m+n+1 \end{cases}$$

Fig. 2. Detailed overview of the inputs and outputs of the modular multiplier in GCM, where $m-1$ is the number of fully filled authenticated data blocks and $n-1$ is the number of fully filled ciphertext blocks. The m^{th} and n^{th} blocks contain the padded data.

B. Threshold Implementation

Masking is delicate to implement, both in software and in hardware. One of the practical difficulties when implementing masking in hardware is that the glitching behavior in ASICs as well as FPGAs may weaken the masked hardware implementation, as shown by Mangard et al. [17], [18]. Thus, a plain Boolean masked implementation in hardware must consider the glitching behavior to minimize leakage. Alternatively, Threshold Implementation (TI) is a masking implementation technique that provides security guarantees even when the underlying hardware glitches [15]. For the recent developments on TI, we refer the reader to the PhD thesis of Bilgin [19].

C. Side-channel Vulnerabilities of AES-GCM

In AES-GCM, both the encryption and the tag generation datapaths must be protected against side-channel attacks. It is straightforward to see that an AES-GCM implementation invoking an unprotected AES as underlying block cipher would result in a design that is vulnerable against DPA. This DPA attack against the block cipher in CTR mode can recover the secret key, resulting in a loss of confidentiality and authenticity. Note that it is reasonably easy to mount DPA attacks on the CTR mode even if the initial counter value is unknown [20].

DPA attacks against the authentication component of AES-GCM are also possible. Textbook DPA does not apply here if the multiplication in the 128-bit field is performed in a single cycle, because the attacker cannot apply a divide-and-conquer strategy. However, other slightly more sophisticated attacks still apply [21], [22]. The outcome of these attacks is the authentication key H. Note that once an adversary learns the authentication key H, he can forge new messages after seeing some ciphertexts, by exploiting the fact that the CTR mode is malleable. Hence, it is important to protect the authentication component of AES-GCM against side-channel attacks as well.

III. Related Work

In 2005, Yang et al. present a throughput-optimized ASIC implementation of AES-GCM in a 0.18 μm standard cell library, achieving a throughput of 34 Gbit/s, running at a frequency of 271 MHz [23]. In 2006, Hodjat et al. concentrate on the optimization of the critical path of the AES cipher in order to maximize the throughput [24]. This results in architectures achieving 30 to 70 Gbit/s in a 0.18 μm standard cell library. In 2007, Lemsitzer et al. present a pipelined implementation of GCM-AES, optimized for FPGAs [25]. Their architecture supports key lengths of 128, 192 and 256 bits. It reaches a throughput of 15.3 Gbit/s on a Virtex-4 FPGA. In the same year, Zhou et al. reach a throughput of 20 Gbit/s by pipelining the AES round and balancing the critical path of the modular multiplier [26]. Improved architectures are presented by Zhou et al. in 2009, resulting in a throughput of 31 Gbit/s and 39 Gbit/s on Virtex-4 and Virtex-5 FPGAs, respectively [27]. The optimizations are applied to the multiplier architecture, resulting in a shorter critical path.

Whereas the aforementioned work concentrates on increasing the throughput, another line of research aims at side-channel security through threshold implementations. In 2011, Moradi et al. present a very compact and threshold implementation of AES [16]. An improved architecture, resulting in a more compact, threshold-protected AES implementation, is presented by Bilgin et al. in [28]. Other approaches include Domain Oriented Masking [29]. All these designs are serialized and do not target high throughput.

In summary, related work covers high-throughput AES and AES-GCM implementations as well as threshold-protected AES implementations on FPGAs and ASICs. Our work concentrates on both high throughput and side-channel protection for AES-GCM on a high-end Virtex-7 FPGA. It is therefore the first high-throughput, side-channel-protected FPGA implementation of AES-GCM.

IV. The Proposed Throughput-optimized and Threshold-protected Architecture

A top-level representation of the implementation is shown in Fig. 3. From left to right there is a counter, the AES threshold component, and the threshold-protected modular multiplier (MALU GMS). The components are glued together through registers, XOR gates and a multiplexer. This section discusses the three components and describes how the architecture works.

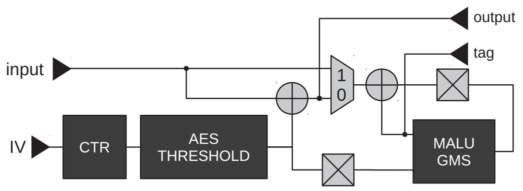


Fig. 3. The proposed top-level architecture

Additionally, it is pointed out that for this implementation, padding is not implemented in hardware, but is expected to be done externally. This implies that the length field, required in the final steps of GCM, is to be provided through the input interface.

Fig. 4 shows a timing diagram that is referred to in the description of the components and the top-level functionality.

A. Components

1) Counter: The counter is initialized with 128 ‘0’ bits for the calculation of H (CTR_RST in Fig. 4). Subsequently, the counter loads the 96-bit initialization vector (IV). For the sake of simplicity, the case where the length of the IV differs from 96 bits is ignored. Upon loading the IV, it is padded with $0x00000002$, which loads $Y1$ into the counter (CTR_LD in Fig. 4). Next, the counter is incremented every clock cycle for each additional plaintext block (CTR_INC in Fig. 4). Finally, the counter is loaded with the IV padded with $0x00000001$, which loads $Y0$ (CTR_CLR in Fig. 4). This comes down to a counter which has two load signals and a reset signal.

The value of $Y0$ is used only after a complete authenticated encryption is done. Because the first value that is used is $Y1$ (which is the incremented version of $Y0$), loading and incrementing the counter takes two clock cycles. This additional clock cycle is not necessary when the second load signal is applied to load $Y1$.

2) AES Threshold: In order to obtain a threshold implementation with three shares, the linear operations in the AES round, namely AddRoundKey, ShiftRows and MixColumns are applied to each of the three shares in parallel. The non-linear operation, i.e. SubBytes, consists of sixteen 8-bit S-box lookups in parallel. A shared implementation of the S-box is achieved by following the approach of Moradi et al. [16].

Depending on the key width, AES runs a number of cycles. This implementation focuses on a 128-bit key, which results in 10 rounds. Instead of implementing one round with a register and having the state cycle through it, this implementation unrolls these 10 rounds. The straightforward way of adding pipelining to the architecture would be to insert a register in between each round. However, the threshold implementation of the S-box requires remasking pipelining registers anyway, as explained in [16]. Therefore, we do not introduce additional pipelining registers in between the rounds. Further, every round needs a round key, which is derived from the original 128-bit key. The key schedule component is therefore also unrolled 10 times, with registers at the same positions inside the S-boxes. Once the pipeline is filled, a new 128-bit encryption is calculated every clock cycle. Because the key schedule is also pipelined, it applies the correct round keys to all the data in the pipeline.

3) MALU GMS: MALU GMS is the threshold-protected Galois Field multiplier, where MALU stands for Modular Arithmetic Logic Unit, as introduced by Sakiyama et al. in [30], and GMS refers to the Generalized Masking Scheme proposed by Reparaz et al. in [31]. Whereas the architecture described in [30] can be configured to different datapath widths, our version uses the full 128-bit width in order to optimize for speed. The single cell and the fully unrolled 128-bit datapath are shown in Fig. 5.

The MALU GMS multiplier uses three shares for each input and three shares for the output. As explained in [31], this leads to first-order DPA protection. If the inputs x and y to the multiplier are shared into x_1, x_2, x_3 and y_1, y_2, y_3, respectively, the shared multiplier first computes a series of cross-products $x_i \cdot y_j$ and then combines them to compress the number of output shares, as defined in [31].

The fully parallel, pipelined AES architecture outputs 128 bits every clock cycle. Therefore, the MALU GMS needs to compute the multiplication result in one cycle as well, in order to be capable of processing a new 128-bit value in each clock cycle.

We note that the multiplications that involve the authenticated data (as shown in Fig. 1) do not need three shares for both inputs, since the authenticated data are not considered to be secret. Therefore, three unprotected Galois Field multipliers could be used in parallel to process the three shares of the

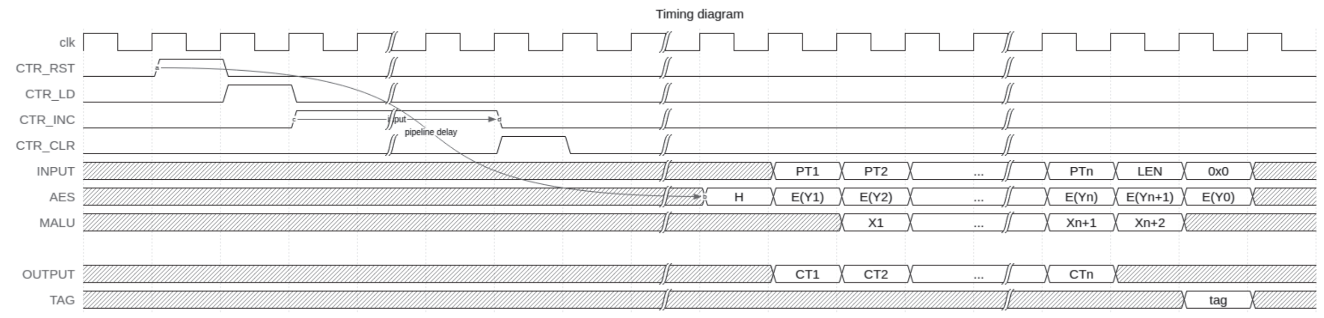


Fig. 4. Sequence of operations in our AES-GCM implementation

other input. Nevertheless, since we re-use the multiplier for calculations in which both operands are secret, both operands are represented in three shares, where the sharing of the authenticated data consists of adding two all-zero masks.

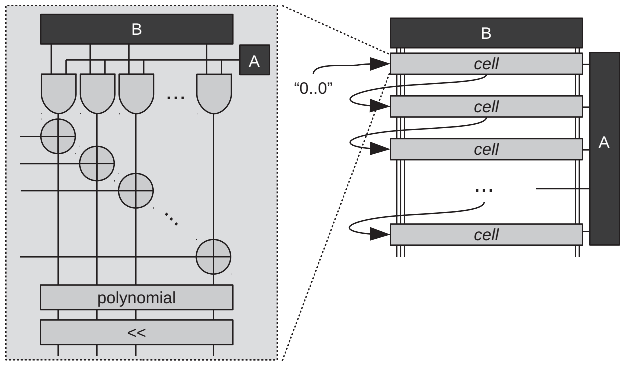


Fig. 5. One MALU cell [30] (left); fully unrolled 128-bit wide MALU (right)

B. Top-level Functionality

The underlying CTR mode for the encryption in GCM uses the block cipher as a stream cipher, which generates a block of key bits rather than a single symbol. Due to the pipelined implementation of the block cipher, the input needs to be fed after the pipeline is filled. Moreover, one additional clock cycle has to be waited for before applying the input, because of the calculation of H.

As described above, GCM uses GMAC for data authentication. The authentication code protects every block of authenticated data and every block of ciphertext. Because the CTR mode behaves as a stream cipher, the ciphertext is generated by XORing the plaintext with the keystream. When decrypting, the ciphertext is again XORed with the keystream, resulting in the plaintext.

When using the implementation for encryption, the multiplexer in Fig. 3 is set to '1' for every block of authenticated data and for the length fields. For every block of plaintext, the multiplexer is set to '1'. When using the implementation for decryption, the multiplexer in Fig. 3 is set to '0' for every block. Hence, for a dedicated decryption implementation, the multiplexer can be removed.

The latency for the computation of the first block of ciphertext is $50 + 1 + m$ clock cycles, where 50 is determined by the number of rounds (10) times the number of pipeline registers; one additional clock cycle for the calculation of H; and where m is the number of blocks of authenticated data. For the computation of the tag, we need to take into account an additional latency of $n + 1 + 1$ clock cycles; where n is the number of plaintext blocks; one additional clock cycle for the MAC update with the length field; and one additional clock cycle for the encryption of $Y0$.

V. Results

The presented architecture is synthesized and implemented using Xilinx' Vivado Design Suite (v2014.4) on a Xilinx Virtex-7 XC7VX485T FPGA. The results of the relevant related work are presented together with our results in Table I.

Table I shows that the throughput of our threshold-protected architecture is in the same order of magnitude as the throughput of previously published unprotected architectures. This is thanks to the comparable clock frequency that can be achieved for our protected architecture on a high-end Virtex-7 device. In terms of configurable resources, the insertion of side-channel countermeasure introduces a large overhead. Note that the slices inside a Virtex-4 FPGA contain 4-to-1-bit Lookup Tables (LUTs), while from Virtex-5 onwards, the slices consist of 6-to-2-bit LUTs.

Typically looking at the filling percentage of an FPGA does not offer additional insights. However, Table I shows that the absolute number of slices between the most recent previous work and this work differs in an order of magnitude. Nevertheless, the position of the Virtex-7 X485T device in the Virtex-7 family is similar to the position of the Virtex-5 LX85 device in the Virtex-5 family. With this in mind, it is clear that with the increasing capabilities of FPGAs, developing side-channel robust implementations is becoming relatively affordable.

The throughput of the proposed architecture is 15.24 Gbit/s. If we assume the most strict requirements on the incoming random numbers, all random masks need to be refreshed every clock cycle. For a single AES round (including key expansion) this is 960 bits per clock cycle for the remasking inside the S-boxes. Further, the plaintext and the key need to be shared,

TABLE I
COMPARISON OF AES-GCM IMPLEMENTATION RESULTS ON FPGA

	FPGA	Slices	Slices relative [%]	BRAM	f$_{max}$ [MHz]	Throughput [Gbit/s]	Side-channel protection
[25][1]	Xilinx Virtex-4 FX100	27'800	66	0	120	15.3	NA
[26][2]	Xilinx Virtex-4 LX40	16'378	88	0	161	20.608	NA
[27][3]	Xilinx Virtex-5 LX85	4'628	36	0	324	41.47	NA
this	Xilinx Virtex-7 X485T	38'241	50	0	119	15.24	1^{st}-order DPA

[1] results for the 128-bit version with the highest throughput
[2] results for the 128-bit speed-efficient encryption
[3] results for the 128-bit highest-throughput encryption

which results in 512 random bits per clock cycle. In total, this comes down to 1472 random bits per clock cycle at 8.4 ns per clock cycle. As a result, if we want to achieve a throughput of 15.24 Gbit/s with the most strict requirement of fresh masks in every clock cycle for first-order DPA protection, the pseudorandom number generator needs to provide random masks at a throughput of 175.24 Gbit/s. This is a throughput that is impossible or, assuming many pseudorandom number generators in parallel, highly impractical to achieve on a Virtex-7 FPGA. Therefore, we conclude that we either need to relax the requirements on the freshness of the masks or reduce the throughput of the AES-GCM architecture, with the additional benefit of the possibility to make the architecture smaller in terms of FPGA resources and to adapt the inputs and outputs to a more commonly used 32-bit or 64-bit interface.

VI. CONCLUSION

In this paper, we investigate the maximum throughput we can achieve for a threshold-protected AES-GCM implementation on a Virtex-7 FPGA. We implement a fully parallel, fully unrolled and pipelined architecture of AES in combination with a Galois Field multiplier, using three input shares and three output shares. The architecture results in a throughput of 15.24 Gbit/s at a clock frequency of 119 MHz. However, if we assume that the most strict requirements are applicable to the generation of random masks, namely fresh masks for each computation in each clock cycle, we need a pseudorandom number generator with a throughput of 175.24 Gbit/s, which is practically infeasible on a Virtex-7 FPGA. Therefore, future work includes the design of an architecture that incorporates a pseudorandom number generator and adapts the throughput of the AES-GCM architecture to the maximum achievable throughput of the pseudorandom number generator. Another option is to thoroughly analyze the security of the system in order to identify if the requirements on the freshness of the random numbers can be relaxed.

ACKNOWLEDGMENT

This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work was supported by the Flemish Government, FWO G.00130.13N, FWO G.0876.14N and Thresholds G0842.13 and by the Hercules Foundation AKUL/11/19.

REFERENCES

[1] NIST, "Recommendation for Block Cipher Modes of Operation: Galois Counter Mode (GCM) and GMAC (SP800-38D)," 2007.
[2] H. Lipmaa, D. Wagner, and P. Rogaway, "Comments to NIST concerning AES modes of operation: CTR-mode encryption," 2000.
[3] J. Daemen and V. Rijmen, *The Design of Rijndael*. Springer-Verlag, 2002.
[4] D. A. McGrew and J. Viega, "The security and performance of the galois/counter mode (GCM) of operation," in *Progress in Cryptology - INDOCRYPT, 5th International Conference on Cryptology in India*, ser. Lecture Notes in Computer Science, A. Canteaut and K. Viswanathan, Eds., vol. 3348. Springer, 2004, pp. 343–355.
[5] "CAESAR: Competition for authenticated encryption: Security, applicability, and robustness." [Online]. Available: https://competitions.cr.yp.to/caesar.html
[6] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference*, 1996, pp. 104–113.
[7] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 388–397.
[8] L. Goubin and J. Patarin, "DES and differential power analysis (the "duplication" method)," in *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99*, ser. Lecture Notes in Computer Science, Ç. K. Koç and C. Paar, Eds., vol. 1717. Springer, 1999, pp. 158–172.
[9] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference*, ser. Lecture Notes in Computer Science, M. J. Wiener, Ed., vol. 1666. Springer, 1999, pp. 398–412.
[10] C. Herbst, E. Oswald, and S. Mangard, "An AES smart card implementation resistant to power analysis attacks," in *Applied Cryptography and Network Security, 4th International Conference, ACNS*, ser. Lecture Notes in Computer Science, J. Zhou, M. Yung, and F. Bao, Eds., vol. 3989, 2006, pp. 239–252.
[11] M. Rivain and E. Prouff, "Provably secure higher-order masking of AES," in *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop*, ser. Lecture Notes in Computer Science, S. Mangard and F. Standaert, Eds., vol. 6225. Springer, 2010, pp. 413–427.
[12] H. Kim, S. Hong, and J. Lim, "A fast and provably secure higher-order masking of AES S-box," in *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop*, ser. Lecture Notes in Computer Science, B. Preneel and T. Takagi, Eds., vol. 6917. Springer, 2011, pp. 95–107.
[13] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC implementation of the AES sboxes," in *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference*, ser. Lecture Notes in Computer Science, B. Preneel, Ed., vol. 2271. Springer, 2002, pp. 67–78.
[14] D. Canright and L. Batina, "A very compact "perfectly masked" S-box for AES," in *Applied Cryptography and Network Security, 6th International Conference, ACNS*, ser. Lecture Notes in Computer Science, S. M. Bellovin, R. Gennaro, A. D. Keromytis, and M. Yung, Eds., vol. 5037, 2008, pp. 446–459.

[15] S. Nikova, V. Rijmen, and M. Schläffer, “Secure hardware implementation of non-linear functions in the presence of glitches,” in *Information Security and Cryptology - ICISC, 11th International Conference*, ser. Lecture Notes in Computer Science, P. J. Lee and J. H. Cheon, Eds., vol. 5461. Springer, 2008, pp. 218–234.

[16] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, “Pushing the limits: A very compact and a threshold implementation of AES,” in *Advances in Cryptology – EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, K. G. Paterson, Ed. Springer, 2011, pp. 69–88.

[17] S. Mangard, T. Popp, and B. M. Gammel, “Side-channel leakage of masked CMOS gates,” in *Topics in Cryptology - CT-RSA 2005, The Cryptographers’ Track at the RSA Conference 2005*, ser. Lecture Notes in Computer Science, A. Menezes, Ed., vol. 3376. Springer, 2005, pp. 351–365.

[18] S. Mangard, N. Pramstaller, and E. Oswald, “Successfully attacking masked AES hardware implementations,” in *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop*, ser. Lecture Notes in Computer Science, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 157–171.

[19] B. Bilgin, “Threshold implementations : as countermeasure against higher-order differential power analysis,” Ph.D. dissertation, University of Twente, Enschede, Netherlands, 2015.

[20] J. Jaffe, “A first-order DPA attack against AES in counter mode with unknown initial counter,” in *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop*, ser. Lecture Notes in Computer Science, P. Paillier and I. Verbauwhede, Eds., vol. 4727. Springer, 2007, pp. 1–13.

[21] S. Belaïd, P. Fouque, and B. Gérard, “Side-channel analysis of multiplications in GF(2128) - application to AES-GCM,” in *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security*, ser. Lecture Notes in Computer Science, P. Sarkar and T. Iwata, Eds., vol. 8874. Springer, 2014, pp. 306–325.

[22] S. Belaïd, J. Coron, P. Fouque, B. Gérard, J. Kammerer, and E. Prouff, “Improved side-channel analysis of finite-field multiplication,” in *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop*, ser. Lecture Notes in Computer Science, T. Güneysu and H. Handschuh, Eds., vol. 9293. Springer, 2015, pp. 395–415.

[23] B. Yang, S. Mishra, and R. Karri, “A high speed architecture for Galois/Counter Mode of operation (GCM),” 2005. [Online]. Available: http://eprint.iacr.org/2005/146

[24] A. Hodjat and I. Verbauwhede, “Area-throughput trade-offs for fully pipelined 30 to 70 gbits/s AES processors,” *IEEE Transactions on Computers*, vol. 55, no. 4, pp. 366–372, 2006.

[25] S. Lemsitzer, J. Wolkerstorfer, N. Felber, and M. Braendli, “Multi-gigabit GCM-AES Architecture Optimized for FPGAs,” in *Cryptographic Hardware and Embedded Systems - CHES 2007: 9th International Workshop*, P. Paillier and I. Verbauwhede, Eds. Springer, 2007, pp. 227–238.

[26] G. Zhou, H. Michalik, and L. Hinsenkamp, “Efficient and high-throughput implementations of AES-GCM on FPGAs,” in *2007 International Conference on Field-Programmable Technology*, 2007, pp. 185–192.

[27] ——, “Improving throughput of AES-GCM with pipelined Karatsuba multipliers on FPGAs,” in *Reconfigurable Computing: Architectures, Tools and Applications: 5th International Workshop, ARC*, 2009, pp. 193–203.

[28] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, “A more efficient AES threshold implementation,” in *Progress in Cryptology – AFRICACRYPT 2014: 7th International Conference on Cryptology in Africa*, D. Pointcheval and D. Vergnaud, Eds. Springer, 2014, pp. 267–284.

[29] H. Gross, S. Mangard, and T. Korak, “An efficient side-channel protected AES implementation with arbitrary protection order,” in *Topics in Cryptology - CT-RSA 2017 - The Cryptographers’ Track at the RSA Conference 2017*, ser. Lecture Notes in Computer Science, H. Handschuh, Ed., vol. 10159. Springer, 2017, pp. 95–112.

[30] K. Sakiyama, L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede, “Small-footprint ALU for public-key processors for pervasive security,” in *Workshop on RFID Security 2006*, ser. Lecture Notes in Computer Science. Springer, 2006, p. 12.

[31] O. Reparaz, B. Bilgin, B. Gierlichs, S. Nikova, and I. Verbauwhede, “Consolidating Masking Schemes,” in *Advances in Cryptology - CRYPTO 2015*, ser. Lecture Notes in Computer Science, R. Gennaro and M. J. B. Robshaw, Eds., vol. 9215. Springer, 2015, pp. 764–783.