High Speed Authenticated Encryption for Slow Changing Key Applications Using Reconfigurable Devices

Karim Moussa Ali Abdellatif, Roselyne Chotin-Avot, Habib Mehrez

To cite this version:

Karim Moussa Ali Abdellatif, Roselyne Chotin-Avot, Habib Mehrez. High Speed Authenticated Encryption for Slow Changing Key Applications Using Reconfigurable Devices. Wireless Days (WD), 2013 IFIP, Nov 2013, Spain. pp.1 - 6. hal-01017858v1

HAL Id: hal-01017858
https://hal.sorbonne-universite.fr/hal-01017858v1

Submitted on 3 Jul 2014 (v1), last revised 11 Jun 2015 (v2)

High Speed Authenticated Encryption for Slow Changing Key Applications Using Reconfigurable Devices

Karim M. Abdellatif, R. Chotin-Avot, and H. Mehrez
LIP6-SoC Laboratory, University of Paris VI, France
{karim.abdellatif, roselyne.chotin-avot, habib.mehrez}@lip6.fr

***Abstract*—Since its acceptance as the adopted authenticated encryption algorithm, AES-GCM has been utilized in various security-constrained applications. This paper describes the benefits of adding key-synthesized property to AES-GCM using FPGAs. Presented architectures can be used for applications which require encryption and authentication with slow changing keys like Virtual Private Networks (VPNs). Three methods are selected to implement the SubBytes of AES to increase the flexibility of the presented work. Furthermore, we propose a protocol to protect the bitstream of the proposed architectures. Our architectures were evaluated using Virtex5 and Virtex4 FPGAs. It is shown that the performance of the presented AES-GCM architectures outperforms the previously reported ones.**

***Keywords*-AES-GCM, FPGAs, VPNs.**

I. Introduction

GCM mode is well-suited for wireless, optical, and magnetic recording systems due to its multi-Gbps authenticated encryption speed, outstanding performance, minimal computational latency as well as high intrinsic degree of pipelining and parallelism. New communication standards like IEEE 802.1ae [1] and NIST 800-38D have considered employing GCM to enhance their performance.

Virtual Private Networks (VPNs) are widely employed to connect private local area networks to remote locations. VPNs use AES-GCM for encryption and authentication. In these kind of networks, the secret key used for encryption and authentication is changed weekly, monthly or yearly. Current commercial security appliances of VPNs allow a throughput from 40 to 60 Gbit/s [2],[3]. Another example of slow changing keys application is embedded system memory protection [4]. This application requires infrequent key changes over weeks or months.

Our contribution: In this work, we present efficient FPGA based architectures for AES-GCM by taking the advantage of slow changing key applications. The key used for encryption and authentication is synthesized into the module structure in order to reduce the consumed area. This is achieved by combining the GF(2^{128}) multiplier proposed by [5] with our presented AES. Moreover, we present a protocol to secure the reconfiguration of the proposed architectures on FPGAs.

Section II presents an introduction and previous work of AES-GCM. After that, our proposed architectures of AES-GCM are presented in **Section III**. Then, we report our implementation results in **Section IV**. **Section V** proposes the protocol which is used for the key exchange and reconfiguration of FPGAs. Finally, **Section VI** concludes our work.

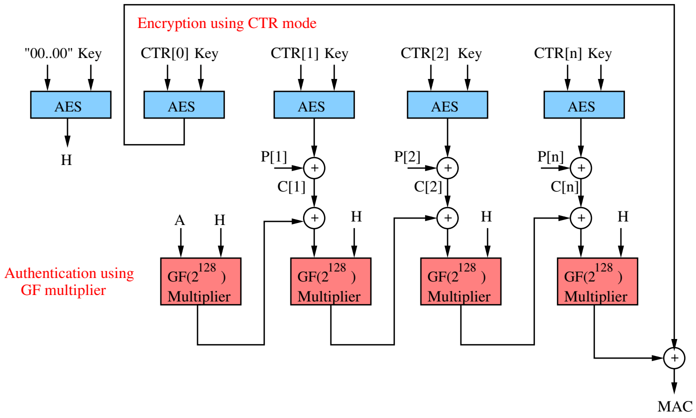


Figure 1. GCM mode of operation

II. AES-GCM

Recently, Galois Counter Mode (GCM)[6] was considered as a new mode of operation of Advanced Encryption Standard (AES). GCM simultaneously provides confidentiality, integrity and authenticity assurances on the data. It supports not only high speed authenticated encryption but also protection against bit-flipping attacks. It can be implemented in hardware to achieve high speeds with low cost and low latency. Software implementations can achieve excellent performance by using table-driven field operations. GCM was designed to meet the need for an authenticated encryption mode that can efficiently achieve speeds of 10 Gbps and higher in hardware. It contains an AES engine in CTR mode and a Galois Hash (GHASH) module as presented in Fig. 1.

As shown in Fig. 1, the GHASH function (authentication part) is composed of chained GF(2^{128}) multipliers and bitwise exclusive-OR (XOR) operations. **Algorithm 1** describes the GF(2^{128}) multiplier. Serial implementation of **Algorithm 1** performs the multiplication process in 128

Algorithm 1: GF(2^{128}) multiplier

```
Input A, H ∈ GF($2^{128}$), F(x) Field Polynomial.
Output X
X=0
for $i = 0$ to 127 do
    if $A_i = 1$ then
        $X \longleftarrow X \oplus H$
    end if
    if $H_{127} = 0$ then
        $H \longleftarrow rightshift(H)$
    else
        $H \longleftarrow rightshift(H) \oplus F(x)$
    end if
end for
return X
```

clock cycles. Parallel method can be implemented like [7] and it takes only one clock cycle.

In **Algorithm 1**, if **H** is fixed, the multiplier is called fixed operand GF(2^{128}) multiplier [5] which can be used efficiently (smaller area) on FPGAs as the circuit is specialized for **H** and a new reconfiguration is uploaded into the FPGA with the new specialization in case of changing the key.

Karatsuba Ofman Algorithm (KOA) was used by [8] to reduce the complexity (consumed area) of GF(2^{128}) multiplier. In order to reduce the data path of KOA multiplier, pipelining concept was accomplished by [9]. However the use of pipelining concept for KOA decreases the data path and increases the operating frequency, the number of clock cycles to process a number of 128-bits is increased. This is because the output is fed back and XORed to the next input as shown in Fig. 2 and there is a latency resulting from pipelining. An example of this problem is shown in [9], their GF multiplier achieves the multiplication of 8 frames of 128-bits in 19 clock cycles because of using the pipelining concept. Their throughput is as follow:

$$Throughput(Mbps) = F_{max(MHz)} \times 128 \times (8/19) \quad (1)$$

Satoh et al. [7] implemented pipelined AES based on composite field approach for SubBytes stage and parallel method for implementing GF(2^{128}) multiplier. Two methods of pipelined AES (composite field and BRAMs) were accomplished with KOA multiplier by [8] using virtex4 FPGA. Zhou et al. [9] used three methods for pipelined AES implementation (composite field, BRAMs, and LUT) with pipelined KOA to increase the operating frequency of the overall architecture. Henzen et al. [10] presented four parallel AES-GCM to support high speed Ethernet applications with using pipelined KOA for GF(2^{128}) and three methods for AES like [9].

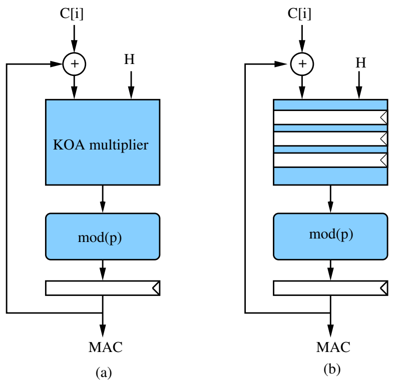


Figure 2. (a) KOA based GHAH; (b) Pipelined KOA based GHASH

III. Efficient AES-GCM for Slow Changing Key Applications Using FPGAs

Applications like VPNs and embedded memory protection are considered slow key changing applications. Therefore, implementing the key expansion is particularly expensive in terms of hardware cost. Also, the GF multiplier used for authentication is a challenge because its data path is longer than AES and pipelining method does not solve this problem as described before.

AES has a key expansion or key schedule operation, which takes the main key and derives from it subkeys K_r (10, 12, and 14 for AES-128, AES-192, and AES-256, respectively), where **r** denotes the corresponding round number. For our case, we concentrate on AES-128.

In our hardware implementation, constant key specialization on the FPGA is used. The precomputed keys are generated using a **C** code as shown in Table I. After, these keys are synthesized into the architecture of AES. As a result, the key expansion scheme is reduced from the architecture of AES.

Table I
Precomputed round keys

Main Key	000102030405060708090a0b0c0d0e0f
Precomputed k0	000102030405060708090a0b0c0d0e0f
Precomputed k1	d6aa74fdd2af72fadaa678f1d6ab76fe
Precomputed k2	b692cf0b643dbdf1be9bc5006830b3fe
Precomputed k3	b6ff744ed2c2c9bf6c590cbf0469bf41
Precomputed k4	47f7f7bc95353e03f96c32bcfd058dfd
Precomputed k5	3caaa3e8a99f9deb50f3af57adf622aa
Precomputed k6	5e390f7df7a69296a7553dc10aa31f6b
Precomputed k7	14f9701ae35fe28c440adf4d4ea9c026
Precomputed k8	47438735a41c65b9e016baf4aebf7ad2
Precomputed k9	549932d1f08557681093ed9cbe2c974e
Precomputed k10	13111d7fe3944a17f307a78b4d2b30c5
Precomputed H	c6a13b37878f5b826f4f8162a1c8d879

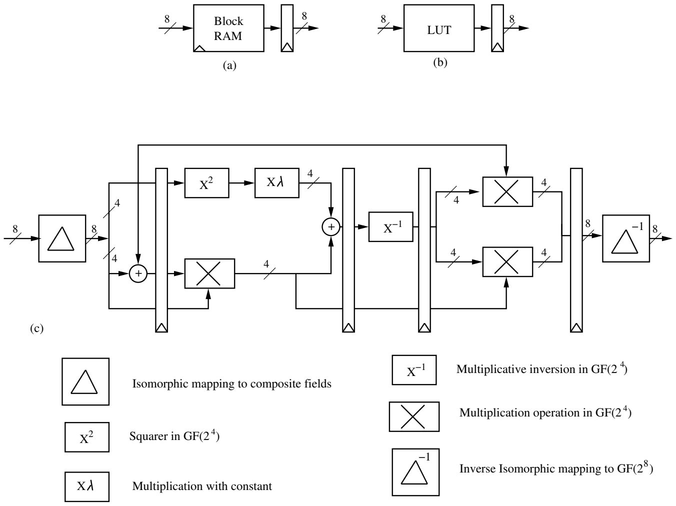

8
Block RAM
8
(a)
8
LUT
8
(b)
8
8
4
4
X^2
$X\lambda$
4
4
X^{-1}
4
4
4
4
8
-1
8
(c)
Isomorphic mapping to composite fields
X^2
Squarer in $GF(2^4)$
$X\lambda$
Multiplication with constant
X^{-1}
Multiplicative inversion in $GF(2^4)$
Multiplication operation in $GF(2^4)$
-1
Inverse Isomorphic mapping to $GF(2^8)$

Figure 3. SubBytes implementation with BlockRAMs (a), with LUTs (b), with composite field approach (c)

The SubBytes transformation can be implemented either by BRAMs, composite field approach or direct LUT approach as shown in Fig. 3. Modern FPGAs contain BRAMs. Therefore, implementing SubBytes using BRAMs decreases the consumed slices of the FPGA. The LUT approach is especially interesting on Virtex5 devices because 6-input Look-Up-Tables (LUT) combined with multiplexors allow an efficient implementation of the AES SubBytes stage. Composite field approach uses the multiplicative inverse of $GF(2^8)$ and it is efficient for memoryless platforms.

As we look for high speed architectures, subpipelining is used to obtain high throughput. Fig. 4 shows synthesized key AES, where all keys are precomputed and synthesized into the architecture.

As a result of using key-synthesized AES, the operand H of the GHASH function is also fixed because it is generated by applying the block cipher to the zero block. Therefore, the proposed multiplier by [5] is suitable because it is based on fixed operand multiplier. Fig. 5 shows the proposed multiplier by [5]. **Algorithm 1** is divided into **Algorithm 2** and **Algorithm 3**. **Algorithm 2** is used to precompute the lookup table based on a fixed H. The lookup table generated by **Algorithm 2** contains 128 vectors of 128 bits. This table is synthesized into the architecture of the multiplier by **Algorithm 3** to compute the $GF(2^{128})$ multiplication. Synthesizing binary 1 values of table T directly perform logic and binary 0 values do not perform logic because of XOR operation as shown in **Algorithm 3**. Therefore, the consumed area is reduced. The overall architecture of AES-GCM is presented in Fig. 6. The proposed architecture limits the logic utilization by specializing the core of AES-GCM on a per key.

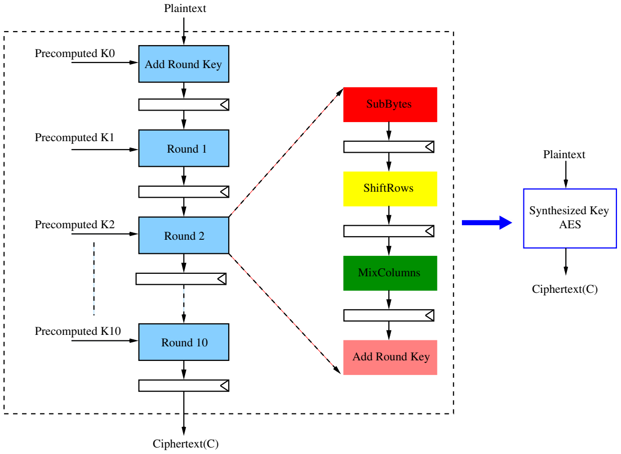
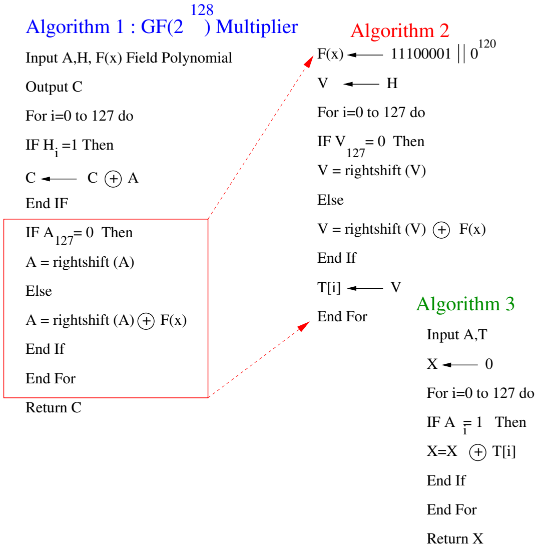


Figure 4. key-synthesized based AES



```
Algorithm 1 : GF(2^128) Multiplier
Input A,H, F(x) Field Polynomial
Output C
For i=0 to 127 do
IF H_i =1 Then
C ← C ⊕ A
End IF
IF A_127= 0 Then
A = rightshift (A)
Else
A = rightshift (A) ⊕ F(x)
End If
End For
Return C
```

```
Algorithm 2
F(x) ← 11100001 || 0^120
V ← H
For i=0 to 127 do
IF V_127 = 0 Then
V = rightshift (V)
Else
V = rightshift (V) ⊕ F(x)
End If
T[i] ← V
End For
```

```
Algorithm 3
Input A,T
X ← 0
For i=0 to 127 do
IF A_i = 1 Then
X=X ⊕ T[i]
End If
End For
Return X
```

Figure 5. $GF(2^{128})$ multiplier proposed by [5]

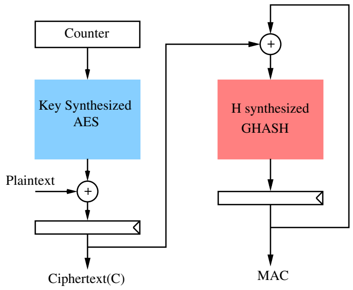


Figure 6. key-synthesized AES-GCM

Table II
Hardware comparison

	FPGA type	Design	Key	SubBytes	Slices	BRAM	Max-Freq MHz	Thr. Gbit/s	Thr./Slice Mbps/Slice
	Virtex4	AES/GCM	○	BRAM	4652	80	216.3	27.7	5.95
o	Virtex4	AES/GCM	○	Comp.	10316	0	239	30.6	2.96
u	Virtex5	AES/GCM	○	BRAM	2478	40	242	30.9	**12.5**
r	Virtex5	AES/GCM	○	Comp.	5512	0	232	29.7	5.38
s	Virtex5	AES/GCM	○	LUT	3211	0	216.3	27.7	8.62
[9]	Virtex4	AES/GCM	●	BRAM	7712	82	285	15.4	1.99
[9]	Virtex4	AES/GCM	●	Comp.	14349	0	277	14.9	1.04
[9]	Virtex5	AES/GCM	●	BRAM	3533	41	314	16.9	4.78
[9]	Virtex5	AES/GCM	●	Comp	6492	0	314	16.9	2.60
[9]	Virtex5	AES/GCM	●	LUT	4628	0	324	17.5	3.77
[8]	Virtex4	AES/GCM	●	Comp.	16378	0	161	20.61	1.26
[11]	Virtex4	AES/GCM	●	BRAM	13200	114	110	14	1.07
[11]	Virtex4	AES/GCM	●	Comp.	21600	0	90	11.52	0.53
[11]	Virtex4	AES/GCM	●	LUT	27800	0	120	15.4	0.55

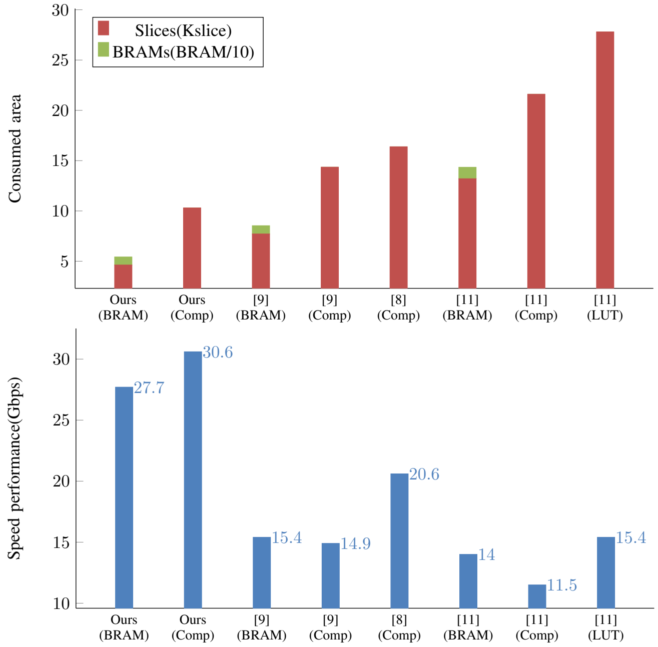


Figure 7. Hardware comparison on Virtex4

IV. Hardware comparison

We coded our proposed architectures in VHDL and targeted to Virtex4 (V4LX60ff668-11) and Virtex5 (XC5VLX220). ModelSim 6.5c was used for simulation. Xilinx Synthesize Technology (XST) is used to perform the synthesize and ISE9.2 was adopted to run the Place And Route (PAR).

Table II shows the hardware comparison between our results and previous work. Note the filled dots in the ”Key” column. Key is synthesized into the architecture when denoted by ○, otherwise, the key schedule is implemented when denoted by ●.

On Virtex4 platform, our key-synthesized based AES-GCM core reaches the throughput of 27.7 Gbps with the area consumption of 4652 slices and 80 BRAMs. In case of using composite field SubBytes, it consumes twice more slices, however no BRAMs are required. Our implementations are technology independent and can be implemented to other FPGA devices. On Virtex5, the most efficient implementation reaches the throughput 30.9 Gbps with 2478 slices and 40 BRAMs.

By comparing our results of AES-GCM to the previous work, the comparison shows that our performance (Thr. /Slice) is better than [8],[9],[11]. The operating frequency presented by [9] is better than ours because they used pipelined KOA but the overall throughput is lower than ours because their GHASH achieves the multiplication of

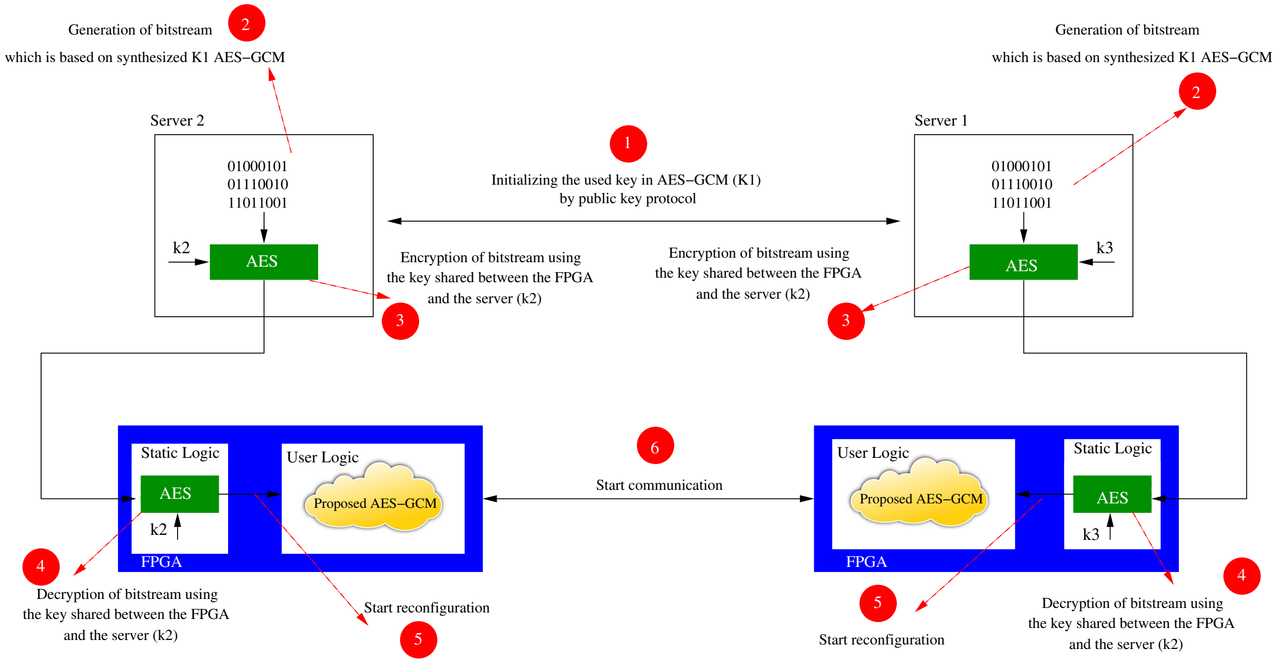


Figure 8. Secure bitstream communication

8 frames of 128-bits in 19 clock cycles. Therefore, their throughput is calculated shown in Eq. 1.

Fig. 7 presents the comparison between our proposed architectures and previous work on Virtex4. It is clear that our work outperforms the previously reported ones. Therefore, proposed architectures can be used efficiently for slow changing key applications like VPNs and embedded memory protection.

V. Bitstream Security of the Proposed Architectures

As a result of synthesizing the key into the architecture, the generated bitstream is key dependent. Therefore, the bitstream must be sent in a secure way to the FPGA. Our analysis focuses on securing the key exchange and the implementation of the dependent-key bitstream on the FPGA. Fig. 8 shows the proposed protocol which is used to perform the key exchange and reconfiguration process between two FPGAs in a secure way. Our scheme assumes that two FPGAs in two different networks are in a communication.

First, the two servers are communicating in order to initialize the key (k1) of AES-GCM. This initialization is performed using public key cryptography. Second, the two servers generate the bitstream file which contains synthesized k1 AES-GCM. Third, the bitstream is encrypted and sent to the FPGA. Thanks to Xilinx because Virtex5 and Virtex4 contain AES engine for supporting secure reconfiguration. Fourth, the two FPGAs decrypt the encrypted bitstream. Fifth, the synthesized k1 AES-GCM is implemented on the user logic. Finally, the communication between the two FPGAs is achieved.

VI. Conclusion

In this paper, we presented the performance improvement of AES-GCM by key-synthesized method. We integrated this concept using three methods of SubBytes implementation. The bitstream of the proposed architectures contains information related to the used key. Hence, we presented a protocol to protect the bitstream of the proposed architectures. Our presented architectures of AES-GCM can be used for slow changing key applications like VPNs and embedded memory protection.

References

[1] "IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Security Amendment 1: Galois Counter Mode–Advanced Encryption Standard– 256 (GCM-AES-256) Cipher Suite," *IEEE*.

[2] C. Corporation, "Cisco ASA 5500 Series Adaptive Security Appliances," 2011. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6032/ps6094/ps6120/prod-brochure0900aecd80285492.pdf.

[3] Stonesoft, "Security Engine Firewall/VPN," 2011. [Online]. Available: http://www.stonesoft.com/export/download/pdf/datasheet-stonesoft-3206.pdf

[4] R. Vaslin, G. Gogniat, J. Diguet, R. Tessier, D. Unnikrishnan, and K. Gaj, "Memory Security Management for Reconfigurable Embedded Systems," *International Conference of Field Programmable Technology (FPT)*, pp. 153–160, 2008.

[5] J. Crenne, P. Cotret, G. Gogniat, R. Tessier, and J. Diguet, “Efficient Key-Dependent Message Authentication in Reconfigurable Hardware,” *International Conference on Field Programmable Technology (FPT)*, pp. 1–6, 2011.

[6] D. McGrew and J. Viega, “The Security and Performance of the Galois/Counter Mode (GCM) of Operation,” *Progress in Cryptology-INDOCRYPT 2004*, pp. 377–413, 2005.

[7] A. Satoh, “High-Speed Hardware Architectures for Authenticated Encryption Mode GCM,” *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 4–pp, 2006.

[8] G. Zhou, H. Michalik, and L. Hinsenkamp, “Efficient and High-Throughput Implementations of AES-GCM on FPGAs,” *International Conference on Field Programmable Technology (FPT)*, pp. 185–192, 2007.

[9] G. Zhou and H. Michalik, “Improving Throughput of AES-GCM with Pipelined Karatsuba Multipliers on FPGAs,” *Reconfigurable Computing: Architectures, Tools and Applications*, pp. 193–203, 2009.

[10] L. Henzen and W. Fichtner, “FPGA Parallel-Pipelined AES-GCM Core for 100G Ethernet Applications,” *Proceedings of the ESSCIRC*, pp. 202–205, 2010.

[11] S. Lemsitzer, J. Wolkerstorfer, N. Felber, and M. Braendli, “Multi-Gigabit GCM-AES Architecture Optimized for FPGAs,” *Cryptographic Hardware and Embedded Systems-CHES*, pp. 227–238, 2007.