Authenticated Encryption on FPGAs from the Static Part to the Reconfigurable Part

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

Authenticated Encryption on FPGAs from the Static Part to the Reconfigurable Part

Karim M. Abdellatif, Roselyne Chotin-Avot, and Habib Mehrez

University of Pierre and Marie Curie, Paris VI, Paris, France
karim.abdellatif@lip6.fr

Abstract

Recently, techniques have been invented to combine encryption and authentication into a single algorithm which is called Authenticated Encryption (AE). Combining these two security services in hardware produces smaller area compared to two separate algorithms.

AE is implemented in the static part of the FPGA (FPGA silicon) in order to secure the reconfiguration process to ensure the confidentiality and integrity of the bitstream. Also, it is used in the reconfigurable part of the FPGA to support applications which need security requirements like Virtual Private Networks (VPNs).

This paper presents two different directions for implementing AE cores on FPGAs. First, we present efficient ASIC implementations of AE algorithms, Counter with Cipher Block Chaining Mode (CCM) and Galois Counter Mode (GCM), which are used in the static part of the FPGA in order to secure the reconfiguration process. Our focus on state of the art algorithms for efficient implementations leads to propose efficient compact architectures in order to be used for FPGA bitstream security. Presented ASIC architectures were evaluated by using 90 and 130 nm technologies. Second, high-throughput GCM architectures are implemented in the reconfigurable part of the FPGA by taking the advantage of slow changing key environments like VPNs and embedded memory protection. The proposed architectures were evaluated using Virtex5 and Virtex4 FPGAs. It is shown that the performance of the presented work outperforms the previously reported ones.

Keywords: FPGAs, authenticated encryption, secure reconfiguration, VPNs.

1. Introduction

FPGAs are essential components to obtain a short design time. They combine the programmability of processors with the performance of custom hardware. Compared to a full custom ASIC design, they are cost efficient, easier to manage and can immediately be put into operation. Furthermore, they can continuously be reprogrammed during and after the design.

The growth of FPGAs applications space has two main implications. Firstly, FPGA designs represent a significant investment that requires protection. Secondly, FPGAs are increasingly being used in applications that require security properties.

In terms of FPGAs designs protection, secure reconfiguration must be performed to ensure the confidentiality and integrity of the bitstream. This is accomplished by adding cryptographic algorithms like the Advanced Encryption Standard (AES) and Message Authentication Code (MAC) in the static part of the FPGA as presented in Virtex-6 FPGAs [12].

Because FPGAs can provide reconfigurability, useful balance between performance, rapid time to market, and flexibility, they have become the implementation target for many critical embedded systems [6, 7, 1].

Our contribution: First, we introduce compact ASIC solutions of AE algorithms, AES-CCM and AES-GCM, for protecting FPGAs designs. The goal of designing efficient compact architectures of AE is to reduce the area of the static part of the FPGA. The developer sends the encrypted bitstream remotely with its Message Authentication Code (MAC) as shown in Fig. 1. The proposed compact AE inserted in the static part decrypts the encrypted bitstream. Also, it computes the MAC and compares it with the bitstream's MAC to ensure the confidentiality and integrity of the bitstream.

Second, this paper also describes the benefits of adding key-synthesized property to AES-GCM using the reconfigurable part (user logic) of the FPGA. Presented architectures can be used for applications which require encryption and authentication with slow changing applications. Also, we present a solution for the parallelization of AES-GCM cores in order to support applications up to 100 Gbps. Our results show that the performance of the presented AES-GCM architectures outperforms the previously reported ones.

The rest of this paper is organized as follows. **Section 2** presents an overview of Authenticated Encryption (AE) algorithms. **Section 3** focuses

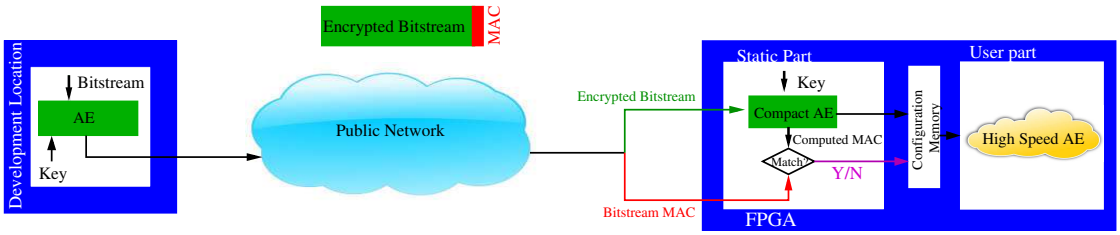

Figure 1: Our contribution

on current solutions of protecting FPGA bitstreams and its drawbacks. **Section 4** proposes efficient compact architectures to be used for FPGA bitstream security. **Section 5** introduces efficient high speed architectures of AES-GCM using FPGAs. **Section 6** concludes this work.

2. Overview of Authenticated Encryption

Previously, confidentiality and authentication services have been implemented separately by using different algorithms. Encryption algorithms are used to ensure confidentiality while Message Authentication Codes (MACs) can be used to provide authentication. When two separate algorithms are used to provide independent security services, it is considered cryptographically secure to use separate keys for each algorithm. Recently, techniques have been invented to combine encryption and authentication into a single algorithm which is called Authenticated Encryption(AE). Combining these two security services in hardware might support the following advantages:

- Area requirement for a single algorithm could be smaller compared to two separate algorithms.
- A slight advantage regarding key management and key storage issues because combined algorithm needs only a single key for both encryption and authentication.

2.1. ***Counter with Cipher Block Chaining-Message Authentication Code (CCM)***

CCM [2] can be used in conjunction with any approved 128-bit block cipher like AES. It is designed for packet environment, where all the necessary data is available in storage before CCM processing. This implies that it is

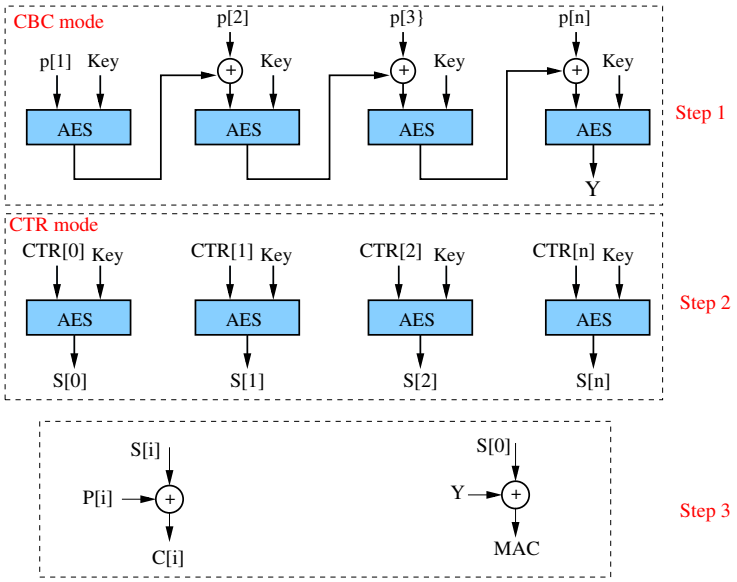

Figure 2: CCM mode of operation

not online. CCM has been specified in the draft IEEE 802.11i standard for wireless networks. It has also been specified in IEEE 802.15 (Wireless Personal Area Networks) and 802.16 (Broadband Wireless Metropolitan Area Networks) standards.

Fig. 2 shows the block diagram of CCM. Firstly, the plaintext P is stored in a memory. Secondly, Y is generated using Cipher Block Chaining mode (CBC mode), this value is used for authentication. Finally, Counter mode (CTR mode) is used to generate ciphered text C. CCM is not suitable for on line applications as all data must be stored in memory before CCM processing.

Another useful feature of CCM mode of operation is handling associated data (i.e. data which must be authenticated but not encrypted. This might be particularly useful in networking applications where data blocks like packet headers are usually sent in the clear, but the receiver must be able to ascertain their source).

In [3], iterative AES (one round) was used to implement the architecture of CCM on FPGA. The proposed architecture in [3] used one AES block for doing authentication and encryption together. Two components of AES were implemented for both encryption and authentication in [4].

2.2. ***Galois Counter Mode (GCM)***

Recently, Galois Counter Mode (GCM)[2] 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. 3.

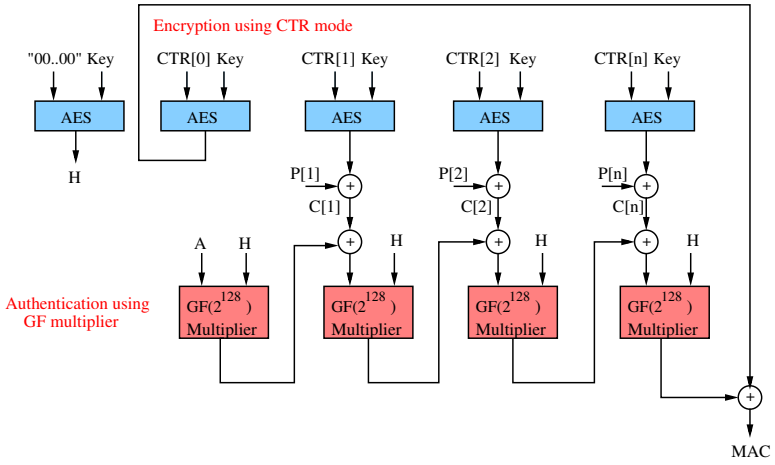

Figure 3: AES-GCM: Encryption process is performed using counter mode and authentication is done using GF(2^{128}), A is an optional 128-bit additional authenticated data which is authenticated but not encrypted

The authentication mechanism within GCM uses GHASH, that features multiplication by the hash subkey, within a binary Galois field. The hash subkey, denoted H, is generated by applying the block cipher to the zero block. GHASH is based on GF(2^{128}) multiplier with irreducible polynomial $F(x) = x^{128} + x^7 + x^2 + x + 1$ as described in [2].

As shown in Fig. 3, the GHASH function (authentication part) is composed of chained GF(2^{128}) multipliers and bitwise exclusive-OR (XOR) operations.

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

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

Algorithm 1 describes the GF(2^{128}) multiplier. Serial implementation of **Algorithm 1** performs the multiplication process in 128 clock cycles. The parallel method uses 128 rounds to achieve the multiplication in one clock cycle and this method is expensive in terms of the consumed area but it is used for high speed applications [5]. In **Algorithm 1**, if **H** is fixed, the multiplier is called fixed operand GF(2^{128}) multiplier [1] 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 [6] to reduce the complexity (consumed area) of GF(2^{128}) multiplier (Fig. 4(a)). In order to reduce the data path of KOA multiplier, pipelining concept was accomplished by [7] (Fig. 4(b)).

Although the use of pipelining concept of 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. 4(b) and there is a latency resulting from pipelining. An example of this problem is shown in [7], their GF(2^{128}) multiplier performed the multiplication of 8 frames of 128-bits in 19 clock cycles because of using the pipelining concept. Their throughput is as follows:

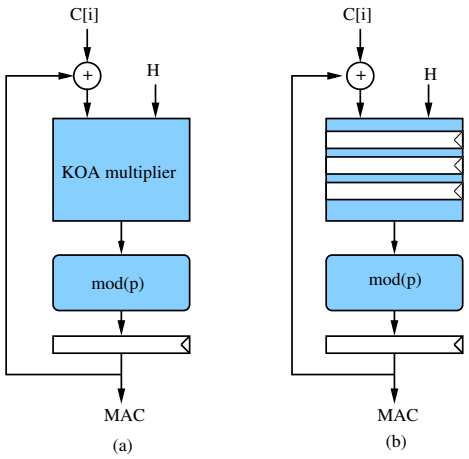

Figure 4: (a) KOA based GHASH; (b) Pipelined KOA based GHASH

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

Two methods of pipelined AES (composite field and Block RAMs (BRAMs)) were accomplished with KOA multiplier by [6] using Virtex4 FPGA. Zhou et al. [7] used pipelined AES with pipelined KOA to increase the operating frequency of the overall architecture. Henzen et al. [8] presented four parallel AES-GCM to support high speed Ethernet applications with using pipelined KOA for GF(2^{128}).

3. Bitstream Security

The main goals and achievements of this section are as follows: (1) to give an overview of security issues used in reconfiguration of FPGAs and analyze how well they are suited to secure the reconfiguration process; (2) to analyze how well encryption and authentication are very important for trusted designs on FPGAs.

3.1. ***Configuration of FPGAs***

In order to redefine the functionality of the FPGA, a *bitstream* configuration file is sent to the FPGA. The bitstream is processed by the static logic- a part of the FPGA that is not programmable in order to establish routing to and from instantiated elements by setting the state of memory cells, pass gates, and routing switches. The user logic is the FPGAs reconfigurable part and where the user-defined application operates.

3.2. ***Remote reconfiguration***

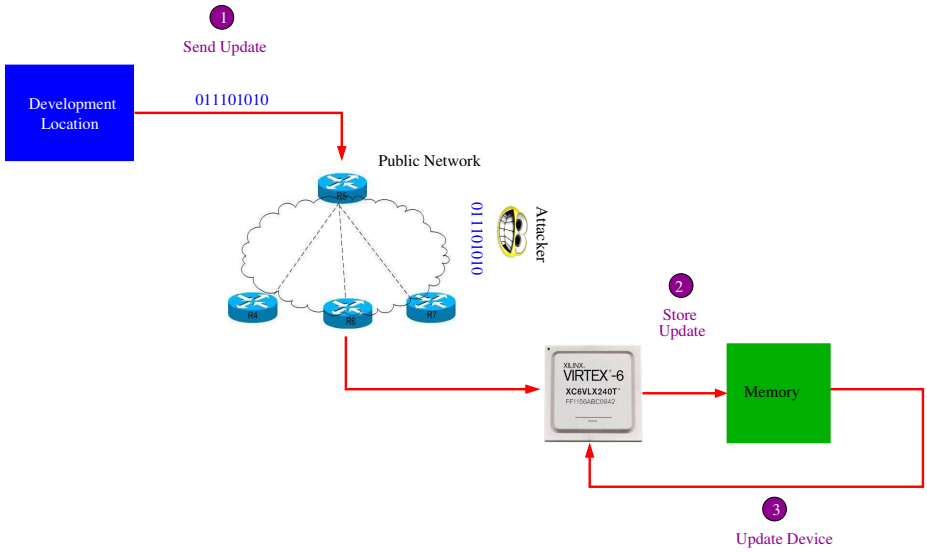

Figure 5: Remote reconfiguration

Reconfiguration of FPGAs is becoming increasingly popular particularly in networking applications and it is vital to provide security against malicious

parties interfering with equipment functionality through this mechanism. Remote reconfiguration is attractive as it is used in such systems to offer new multimedia features or to repair eventual security issues. It requires transmitting the bitstream file which contains the hardware Intellectual Property (IP) over insecure communication channels and thus introduces new security issues as shown in Fig. 5.

The developer is faced with several problems resulting from sending the bitstream file through insecure network. An adversary attacker can detect the hardware IP to sell illegal copies or leak it to the public domain.

*3.3. **Previous solutions***

- **Bitstream confidentiality**:
  In order to secure the bitstream file and prevent attacking, encryption is used. Encryption provides data confidentiality and privacy. FPGAs include hardwired mechanisms that ensure bitstream confidentiality [9]. Bitstream encryption, first introduced by Xilinx on a production level with Virtex II FPGAs to prevent device cloning and to protect the confidentiality of the design data. Each Virtex-4, Virtex-5, and Virtex-6 device have an on-chip AES [10] decryption engine to support encrypted bitstreams. The bitstream is encrypted with a symmetric key K shared between the FPGA circuit and the developer. Key setup is performed in a secure area by the developer before the system is shipped. The encrypted bitstream is decrypted using the static logic as shown in Fig. 6. This mechanism allows for protection of the system designer's IP against cloning as well as reverse engineering.

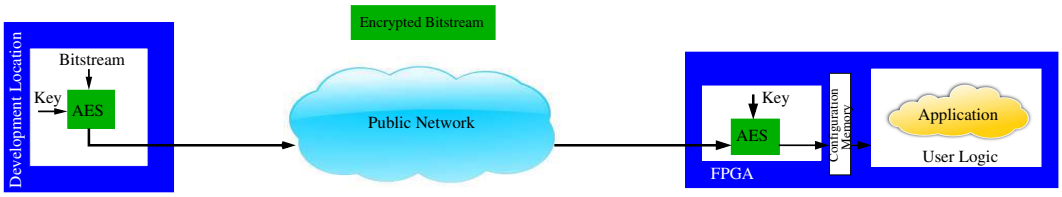

Figure 6: Bitstream encryption

This behavior is not enough to prevent attackers from destroying the FPGA remotely using certain malicious bit-stream combinations. Therefore, the FPGA should accept only bitstreams from an authenticated source.

- **Bitstream integrity**:
  Tampering attack is based on the modification of the bitstream. Therefore, the FPGA must be smart enough to detect the concept of **Who is the sender?**, to accept the correct bitstream sent by the trusted sender. Some FPGA vendors implement Cyclic Redundancy Checks (CRC) [11]. However, the purpose of CRC is to detect transmission errors, not to check the integrity of data in the cryptographic sense. This is why Xilinx [12] suggested using MAC algorithm to ensure the integrity of the bitstream. Virtex-6 FPGAs are the first (and only) programmable devices to offer cryptographically strong bitstream authentication. An on-chip bitstream keyed-MAC algorithm implemented in hardware provides additional security beyond that of using AES bitstream encryption alone [12]. Fig. 7 shows the architecture used in Virtex-6. This concept allows for protection of the system designers IP against tampering attack.

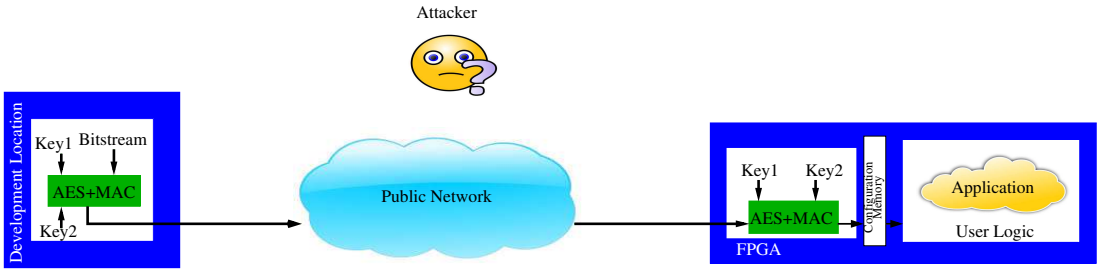

Figure 7: Bitstream encryption and authentication in Virtex6

Parelkar [3] noted that generic composition of authentication and encryption (AES+MAC) required more circuit area than authenticated encryption (AE) algorithms. The advantages of using one algorithm for both encryption and authentication are: smaller area and one key is used for encryption and authentication. Therefore, Parelkar [3] recommended counter with cipher block chaining-message(CCM) mode for achieving both authentication and encryption. Table 1 shows the difference between the hardware implementation of CCM mode and AES with MAC. It is clear that CCM needs smaller area than (AES+MAC).

Table 1: Hardware comparison

Design	architecture	Technology	Area mm^2	Frequency MHz	Throughput Mbps
Parelkar et al.[3]	AES-CCM	90 nm	0.057	148	434
Parelkar et al.[3]	AES+HMAC	90 nm	0.183	101.2	1293

4. Proposed Efficient Compact Architectures for Bitstream Security

Our goal is to design an efficient compact solution in order to be used for encryption and authentication of bitstream. The reason of compact solution is to reduce the used area of the static part which is responsible for the security task. Efficient hardware implementations of AE is presented and compared with previous work. Presented architectures include AES-CCM and AES-GCM.

4.1. ***Compact CCM***

In the proposed AES-CCM, we use 32-bit AES (1/4 round) that has an advantage of reducing the consumed area with a suitable throughput that is able to support applications lower than 1Gbps. Fig. 8 shows the architecture of 32-bit AES. The key schedule shares the SubBytes stage with the data bus. As a result, there are only four s-boxes used. Therefore, we can avoid the long data path resulting from using composite field approach by implementing a ROM to store the values of s-boxes. Moreover, only one MixColumn stage is used. Processing a frame of 128-bit in 1/4 round AES takes 5 clock cycles (four cycles for the data and one cycle for the key). Therefore, achieving the encryption takes 55 clock cycles ($5 \times 11 = 55$) because of 11 round AES.

Our AES-CCM architecture uses one 32-bit AES (1/4 round) for both encryption and authentication as shown in Fig. 9. All data must be stored in a memory. Firstly, authentication process is accomplished using CBC mode. Secondly, encryption process is performed using CTR mode. A 128-bit frame takes 55 clock cycles to be encrypted or added to MAC queue. The achieved throughput of our presented AES-CCM is calculated as follows:

$$Throughput(Mbps) = \frac{128 \times F_{max(MHz)}}{55 \times 2} \tag{2}$$

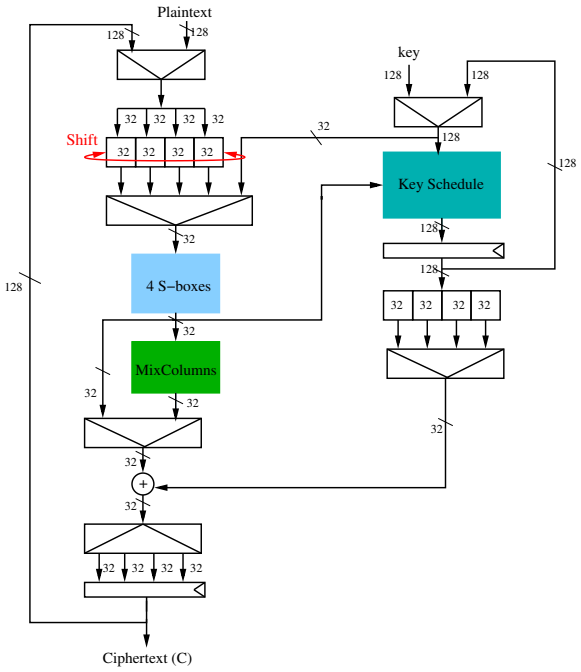

Figure 8: 32-bit AES

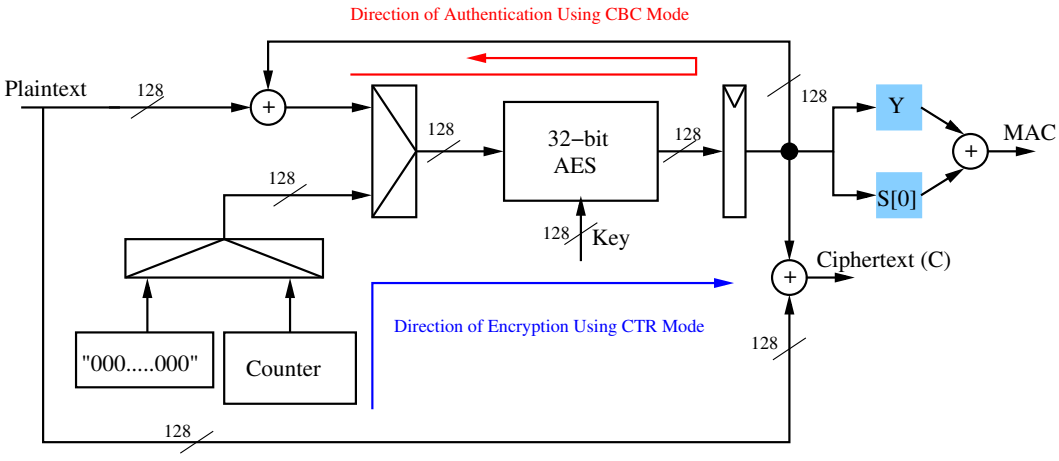

Figure 9: Proposed CCM

4.2. ***Proposed GCM***

AES-GCM uses two components: an AES engine and a GF(2^{128}) multiplier. The target must be directed to optimize the overall architecture which

includes the encryption part (AES) and the authentication part (GF(2^{128})).

Our proposed architecture uses 32-bit AES for area optimization. It performs the encryption in 55 clock cycles as described before. Therefore, the GF(2^{128}) multiplier has to compute the multiplication process in 55 clock cycles or less in order to keep up with 32-bit AES. Previous architectures of GF(2^{128}) like [5, 16] were used for high speed applications. Their architectures achieved the multiplication process in one clock cycle and considered very cost to be used with 32-bit AES. Hence, it is important to design a multiplier which can be used efficiently with 32-bit AES.

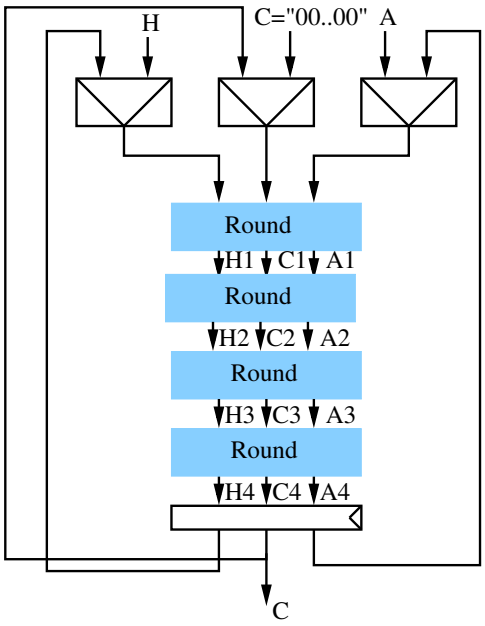

Figure 10: Proposed GF(2^{128}) multiplier

Serial GF(2^{128}) multiplier is described in **Algorithm 1** [2], where A,H are inputs to the multiplier and $F(x)$ is the field polynomial, $F(x) = x^{128} + x^7 + x^2 + x + 1$. The output C needs 128 clock cycles to be ready in case of using serial multiplier. As shown in **Algorithm 1**, it is possible to design one round. Four rounds are used together (hybrid multiplier) for reducing the number of clock cycles needed to perform the multiplication from 128 to 32 (128/4) clock cycles as shown in Fig. 10. This method is very compact compared to [5, 16] because we used only four rounds rather than 128 rounds.

Our proposed AES-GCM architecture shown in Fig. 11 uses 32-bit AES with the hybrid GF(2^{128}) multiplier to accomplish the task of encryption and authentication, respectively. First, the input to 32-bit AES block is zero's to perform H for the GF multiplier. Second, AES changes its mode to be in CTR mode for encryption while GF multiplier performs authentication task. The throughput of the proposed AES-GCM is as follows:

$$Throughput(Mbps) = \frac{128 \times F_{max(MHz)}}{55} \tag{3}$$

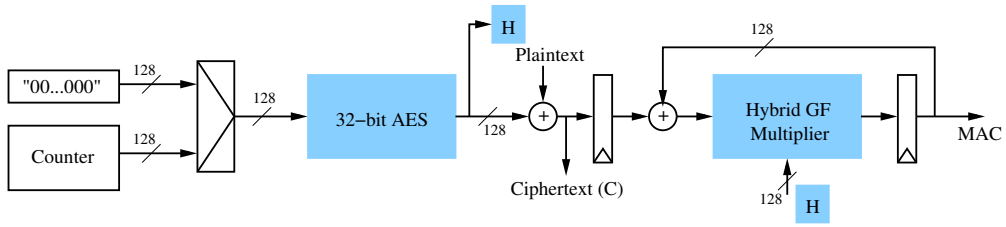

Figure 11: Proposed GCM architecture

4.3. ***Hardware Comparison***

This section compares our presented architectures with the previous work. Presented architectures have been evaluated using 90 and 130 nm CMOS standard cell library and its performances are compared with the prior art in Table 2.

Table 2: Hardware comparison

Design	Architecture	Technology	Area mm^2	Frequency MHz	Throughput Mbps	Performance Gbps/mm^2
This work	AES-CCM	130 nm	0.1407	285	331.6	2.36
This work	AES-GCM	130 nm	0.1615	285	663.2	4.1
This work	AES-CCM	90 nm	0.045	350	407.2	**9**
This work	AES-GCM	90 nm	0.064	344	800.5	**12.5**
[15]	AES	110 nm	0.099	222.2	526.74	5.32
[3]	AES-CCM	90 nm	0.057	148	434	7.6
[3]	AES+HMAC	90 nm	0.183	101.2	1293	7

The reason of using two different libraries (90 and 130 nm) in evaluating our proposed architectures is to present different estimations for the consumed area. In terms of using 90nm technology, the proposed CCM occupies 0.045 mm^2 with 350 MHz as a maximum frequency and GCM needs 0.064 mm^2 with operating frequency 344 MHz.

Although Parelkar et al. [3] presented one architecture of 128-bit AES for both encryption and authentication, our AES-CCM consumes 21 % smaller area than [3] because we used 32-bit AES which has only four s-boxes. The overall performance of our AES-CCM (Throughput/mm^2) is better than [3].

In terms of our AES-GCM mode, our proposed architecture presents smaller consumed area compared to AES+HMAC by [3] (65 % smaller area). In total, the overall performance of our presented AES-GCM is better than AES+HMAC by [3].

In [15], a 32-bit AES was proposed and used only for encryption. However, our architectures (AES-CCM and AES-GCM) perform encryption and authentication together and present better performance (Throughput/mm^2).

4.4. ***Proposed architectures for Bitstream Security***

This section describes how our efficient compact architectures can be used for bitstream security. AE is used in the static part of the FPGA as shown in Fig. 12. The encrypted bitstream is decrypted using AE. Also, AE is used to compute the MAC and compare it with the bitstream's MAC. If they are equal, the FPGA will continue to the startup sequence. Otherwise, configuration will abort and the cells be cleared.

Unlike current FPGAs [17],[18] which support only encryption for bitstream security, our efficient compact solutions add encryption and authentication in order to enhance the security of the configuration process. Moreover, proposed solutions meet the encryption speed of current devices as shown in Table 3.

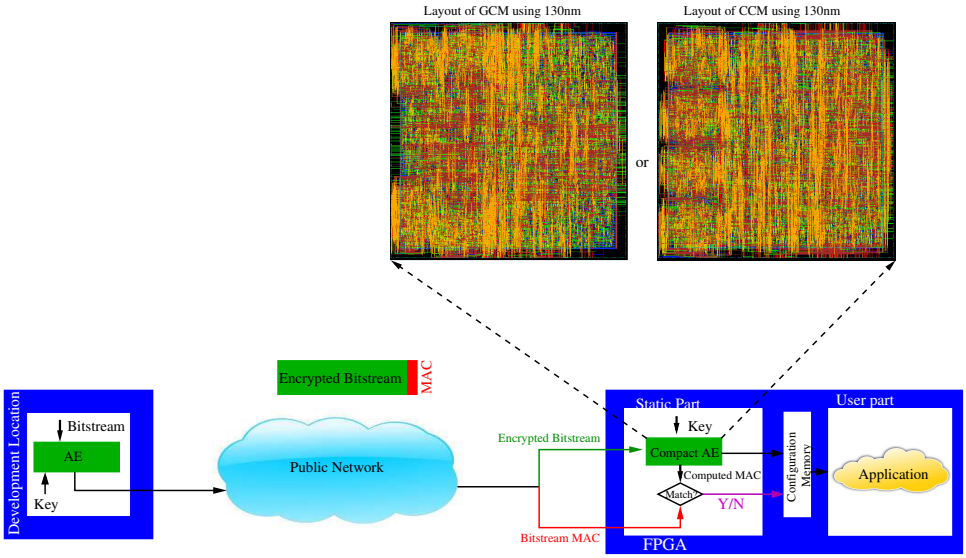

Figure 12: Bitstream security using proposed AE architectures

Table 3: Configuration throughput of some FPGA family members

FPGA	device	Technology	Throughput
Virtex-5[19]	LX330T	65-nm	800 Mbits/s
Spartan-3 [20]	5000	90-nm	400 Mbits/s

5. High Speed GCM Architectures Using FPGAs

Virtual Private Networks (VPNs) are widely employed to connect private local area networks to remote locations. VPNs use AES-GCM for encryption and authentication. The secret key used for encryption and authentication in these networks is changed weekly, monthly or yearly. Current commercial security appliances of VPNs allow a throughput from 40 to 60 Gbps [21],[22]. Another example of slow changing keys application is embedded system memory protection [23].

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, specializing the associated circuitry and reducing module area. In addition, we present a protocol to secure the reconfiguration of these architectures on FPGAs.

5.1. ***Key-synthesized AES-GCM***

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.

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.

In our new hardware implementation, constant key specialization in the FPGA is used. The precomputed keys are generated using a **C** code as shown in Table 4. After, these keys are synthesized into the architecture of AES. As a result, the key expansion scheme is reduced from the architecture of AES.

The SubBytes transformation can be implemented either by BRAMs, composite field or direct Look Up Tables (LUT). Modern FPGAs contain BlockRAMs. 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 shown in Fig. 13.

Table 4: 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

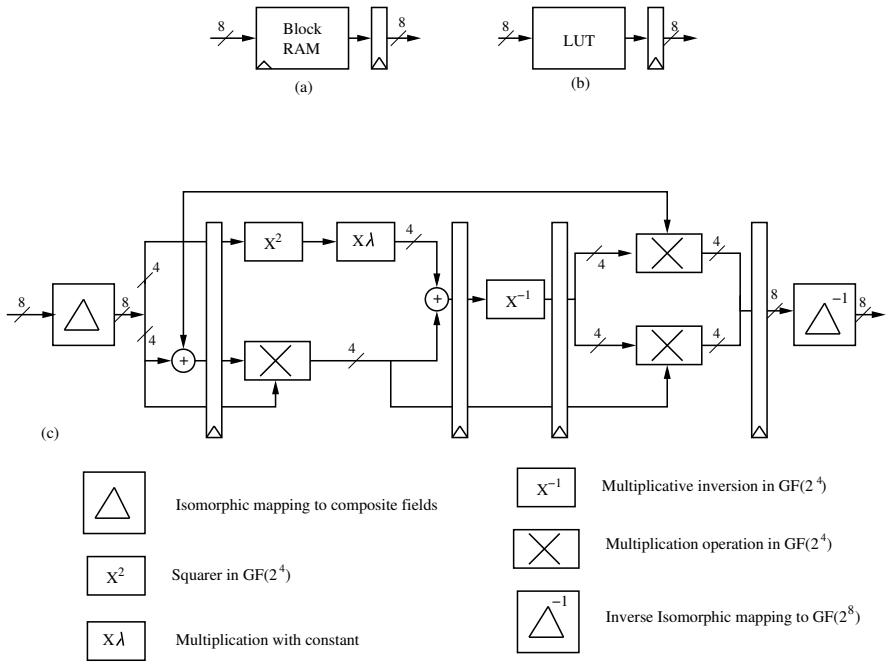

Figure 13: SubBytes implementation with BlockRAMs (a), with LUTs (b), with composite field approach (c)

As we look for high speed architectures, subpipelining is used to obtain high throughput. Fig. 14 shows key-synthesized AES, where all keys are precomputed and synthesized into the architecture.

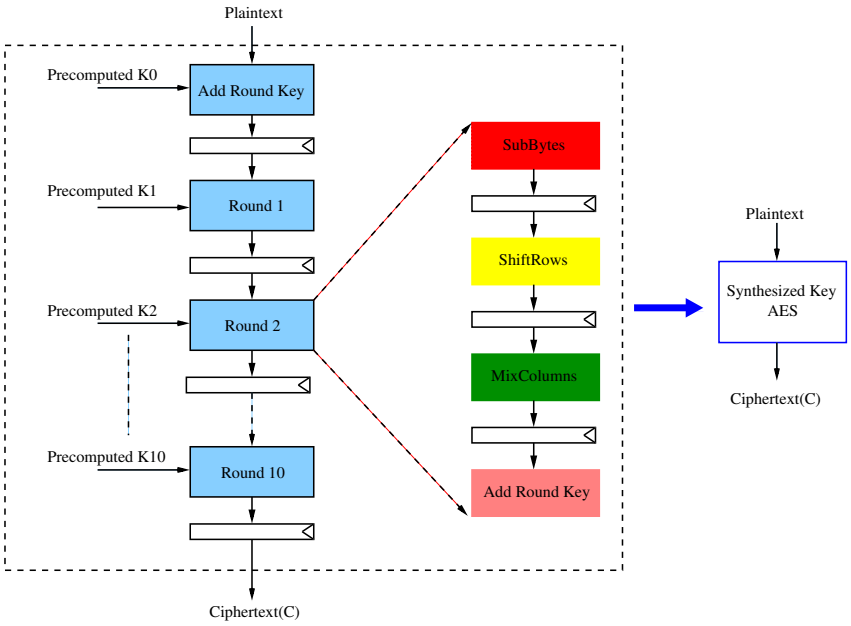

Figure 14: Key-synthesized based AES

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 [1] is very suitable because it is based on fixed operand multiplier. In [1], **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. 16. The proposed architecture limits the logic utilization by specializing the core of

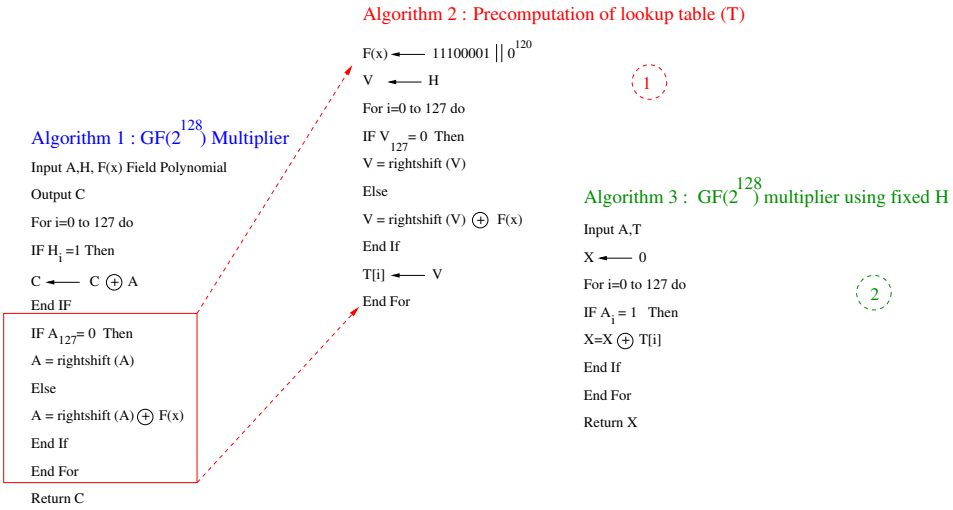

Figure 15: GF(2^{128}) multiplier proposed by [1]

AES-GCM on a per key. VPNs infrastructure can benefit from our key-synthesized AES-GCM implementation [21] due to the nature of slow changing key operation. The next section describes using parallel AES-GCM with key-synthesizing approach.

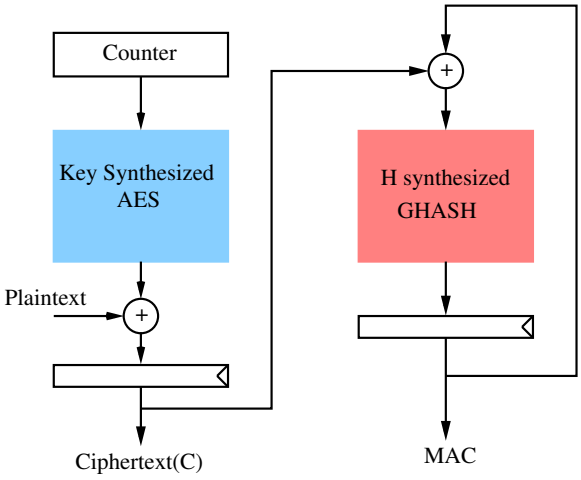

Figure 16: Key-synthesized based-AES-GCM

5.2. ***Parallel Key-synthesized AES-GCM***

In order to implement parallel architectures of AES-GCM using key synthesized method, parallel GHASHs must be constructed to meet the requirement of key-synthesized method (i.e, one of the two operands of each GHASH must be fixed).

Previous parallel schemes of GHASH [16, 24] are not suitable because the two operands of each GHASH are varied during the running time operation. As a result, their architectures are not suitable for key-synthesized approach. Therefore, constructing parallel GHASHs which have a fixed operand for each GHASH multiplier is very important for high speed applications.

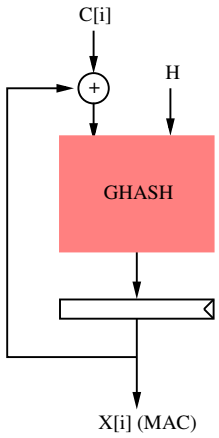

Figure 17: GHASH operation

Fig. 17 shows the GF(2^{128}) multiplication between **H** and 128-bit input value **C**. $GHASH_H$ function for block i is defined in eq. 4.

$$X_i = (C_i \oplus X_{i-1}) \times H \tag{4}$$

In order to accomplish parallel architectures for higher throughput, eq. 4 is modified to fit the parallel scheme as shown in eq. 5. For each multiplier, there is a fixed operand as shown in Fig. 18. For example, $Multiplier_i$ has H as an operand, $Multiplier_{i-1}$ has H^2, ..., and $Multiplier_1$ has H^i .

$$
\begin{aligned}
X_i &= (C_i \oplus X_{i-1}) \times H \\
&= (C_i \times H) \oplus (X_{i-1} \times H) \\
&= (C_i \times H) \oplus [(C_{i-1} \oplus X_{i-2}) \times H^2] \\
&= (C_i \times H) \oplus (C_{i-1} \times H^2) \oplus [(C_{i-2} \oplus X_{i-3}) \times H^3] \\
&= (C_i \times H) \oplus (C_{i-1} \times H^2) \oplus (C_{i-2} \times H^3) \\
&\oplus [(C_{i-3} \oplus X_{i-4}) \times H^4] \\
&= \underbrace{(C_i \times H)} \oplus \underbrace{(C_{i-1} \times H^2)} \oplus \underbrace{(C_{i-2} \times H^3)} \\
&\oplus \underbrace{(C_{i-3} \times H^4)} .... \oplus \underbrace{(C_2 \times H^{i-1})} \oplus \underbrace{(C_1 \times H^i)}
\end{aligned} \tag{5}
$$

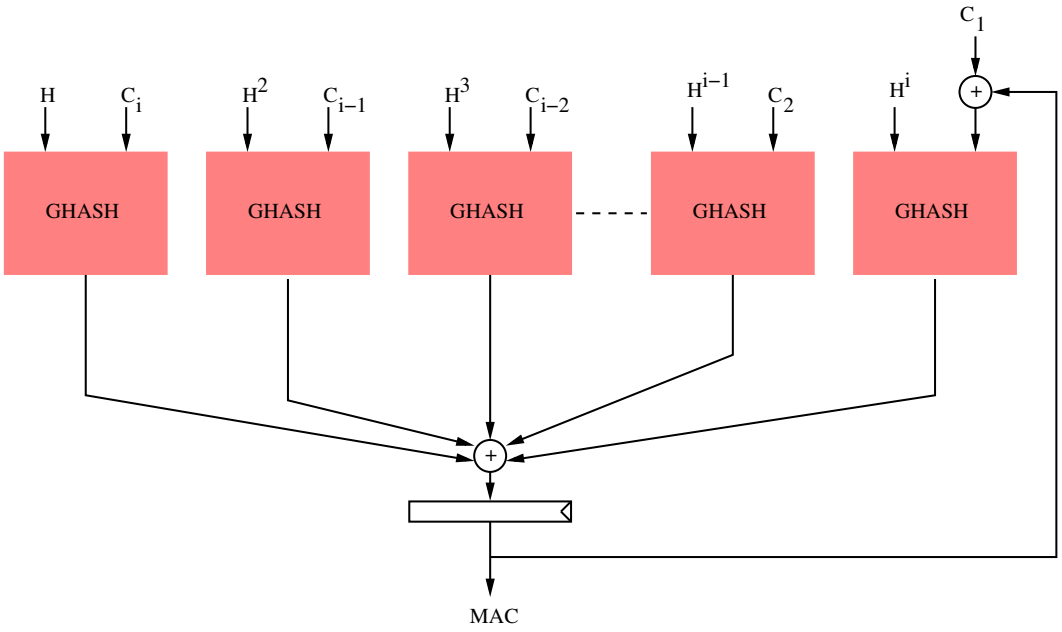

Figure 18: Proposed parallel GHASH with fixed operand during running time operation

Unlike previous work, this method makes the parallel architecture of GHASH suitable for key-synthesized method as we can get these values (H^i, H^{i-1}, ... H) synthesized into the architecture of the parallel GHASH.

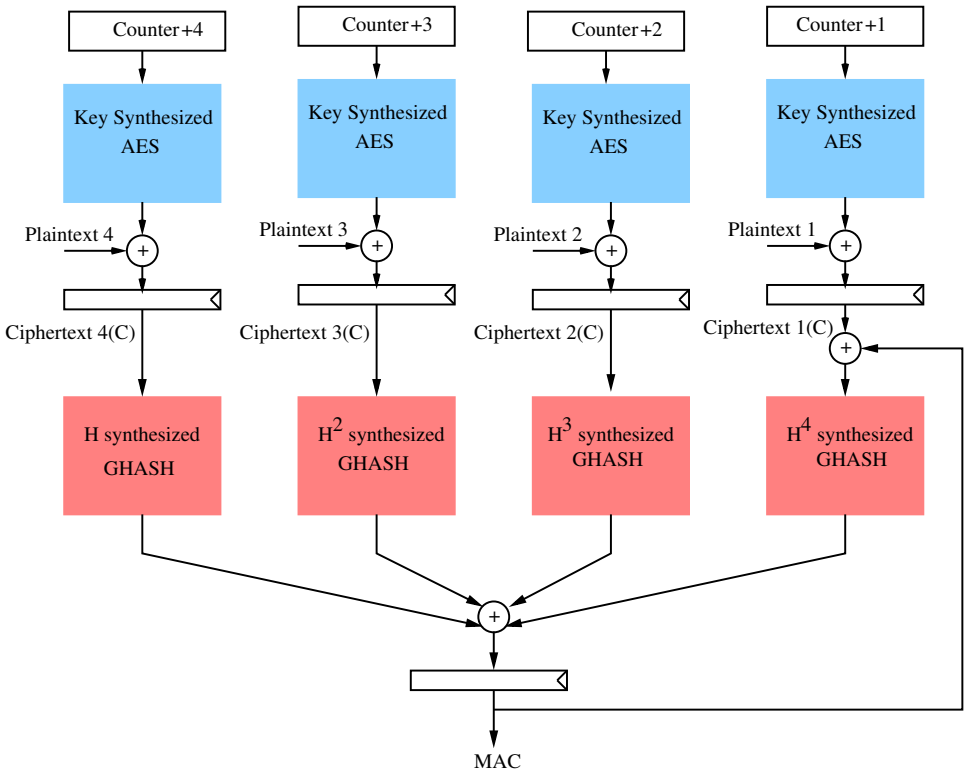

Figure 19: Presented 4-parallel AES-GCM using key-synthesized method

Fig. 19 shows an example of 4-parallel AES-GCM architecture using key-synthesized method. For parallel pipelined AES, the round keys are precomputed and synthesized into the architecture. In terms of parallel the GHASH, the operands (H, H^2, H^3, H^4) are also precomputed and synthesized into the architecture.

As mentioned earlier, not all AES-GCM applications are suitable for our approach because our idea is suitable for slow changing key applications like VPNs.

5.3. Hardware comparison

We coded our proposed schemes (AES-GCM and 4-parallel AES-GCM) 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 5 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. On Virtex5, the most efficient implementation reaches the throughput 30.9 Gbps with 2478 slices and 40 BRAMs. Our implementations are technology independent and can be implemented to other FPGA devices.

By comparing our results of AES-GCM to the previous work, the comparison shows that our performance (Thr. /Slice) is better than [6],[7],[25]. The operating frequency presented by [7] is better than ours because they used pipelined KOA but the overall throughput is lower than ours because their GHASH achieves the multiplication of 8 frames of 128-bits in 19 clock cycles. Therefore, their throughput is calculated as:

$$Throughput(Mbps) = F_{max(MHz)} \times 128 \times \frac{8}{19} \tag{6}$$

We motivate using LUT method for parallel AES-GCM in case of using Virtex-5 to avoid the limit of BRAMs. A 4-parallel AES-GCM module operates at 200 MHz on Virtex-5. In total, it consumes 12152 Slices. This implementation achieves throughput that reaches to 102.4 Gbps. Henzen et al. [8] proposed 4-parallel AES-GCM using pipelined KOA. Their design achieves the authentication of 18 frames of 128-bits in 11 clock cycles because of the latency resulting from the pipelined KOA. As a result, their throughput is calculated as follows:

$$Throughput(Mbps) = F_{max(MHz)} \times 128 \times \frac{18}{11} \tag{7}$$

Fig. 20 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.

Table 5: 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
	Virtex5	4-parallel AES-GCM	○	LUT	12152	0	200	102.4	8.42
[7]	Virtex4	AES-GCM	•	BRAM	7712	82	285	15.4	1.99
[7]	Virtex4	AES-GCM	•	Comp.	14349	0	277	14.9	1.04
[7]	Virtex5	AES-GCM	•	BRAM	3533	41	314	16.9	4.78
[7]	Virtex5	AES-GCM	•	Comp	6492	0	314	16.9	2.60
[7]	Virtex5	AES-GCM	•	LUT	4628	0	324	17.5	3.77
[6]	Virtex4	AES-GCM	•	Comp.	16378	0	161	20.61	1.26
[25]	Virtex4	AES-GCM	•	BRAM	13200	114	110	14	1.07
[25]	Virtex4	AES-GCM	•	Comp.	21600	0	90	11.52	0.53
[8]	Virtex5	4-parallel AES-GCM	•	LUT	14799	0	233	48.8	3.29

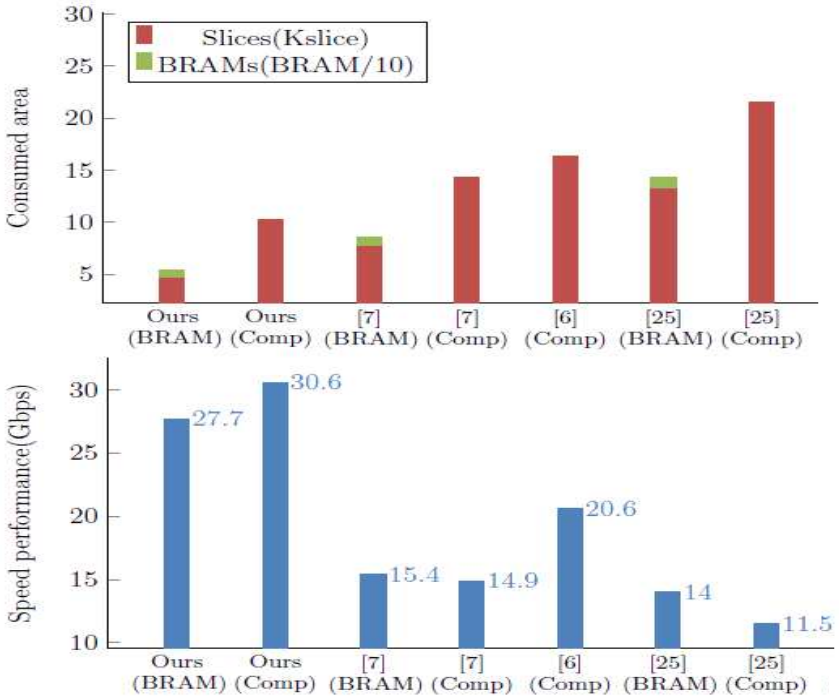

Figure 20: Hardware comparison on Virtex4

5.4. ***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 key-dependent bitstream on the FPGA.

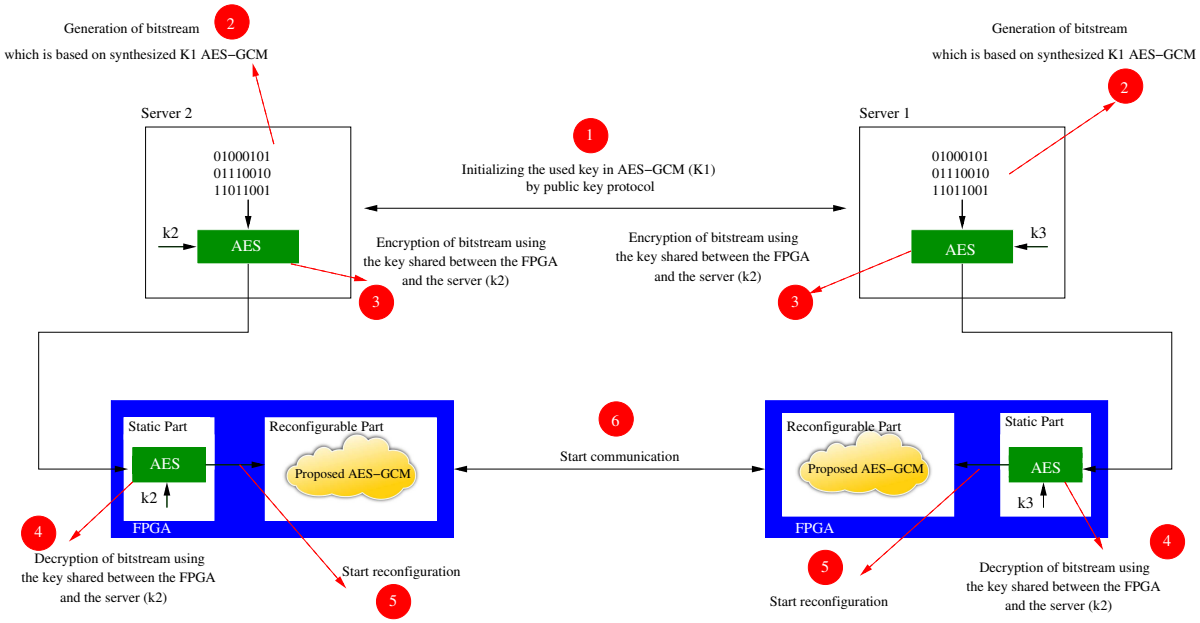

Figure 21: Secure bitstream communication

Fig. 21 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 an AES engine for supporting secure reconfiguration, in case of Virtex6, the bitstram can be also authenticated because Virtex6 has an on chip MAC for supporting authentication. 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.

6. Conclusion

This paper presented two different approaches to implement AE cores on FPGAs. The first approach aims at designing efficient compact ASIC architectures for AE algorithms, AES-CCM and AES-GCM, for protecting FPGAs bitstream. The reason of compact architectures is to reduce the static part of the FPGA which is responsible for bitstream security. These compact architectures are added in the static part of the FPGA in order to decrypt and authenticate the bitstream. Presented ASIC architectures were evaluated by using 90 and 130 nm technologies. Our comparison to previous work reveals that our architectures are more resource-efficient. The second approach includes implementing high speed AES-GCM architectures using the reconfigurable (user logic) part of the FPGA. We presented the performance improvement of AES-GCM by key-synthesized method. With our proposed parallel AES-GCM, each multiplier has a fixed operand. Therefore, presented parallel AES-GCM is suitable for key-synthesized method. Our presented FPGA based architectures of AES-GCM can be used for slow changing key applications like VPNs and embedded memory protection. Our results show that the performance of the presented AES-GCM architectures outperforms the previously reported ones. Future work includes studying the effect of side channel attacks on the proposed architectures.

References

[1] 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.

[2] D. McGrew and J. Viega , "The security and performance of the Galois/Counter Mode (GCM) of operation", Progress in Cryptology-INDOCRYPT, pp. 377–413, 2005.

[3] M. Parelkar, "Authenticated Encryption in Hardware", PhD thesis, George Mason University, 2005.

[4] E. Lopez-Trejo and F. Henriquez, "An Efficient FPGA Implementation of CCM Mode Using AES", International Conference on Information Security and Cryptology, pp. 208–215, 2005.

[5] A. Satoh, "High-speed hardware architectures for authenticated encryption mode GCM", IEEE International Symposium on Circuits and Systems,ISCAS , year=2006.

[6] 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.

[7] G. Zhou, H. Michalik, and L. Hinsenkamp, "Improving Throughput of AES-GCM with Pipelined Karatsuba Multipliers on FPGAs", Journal of Reconfigurable Computing: Architectures, Tools and Applications, pp. 193-203, 2009.

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

[9] A. Lesea, "IP security in FPGAs", `http://direct.xilinx.com/bvdocs/whitepapers/wp261.pdf`, 2007.

[10] NIST FIPS, "197: Advanced encryption standard (AES)", Federal Information Processing Standards Publication, volume=197, pp. 441–0311, 2001.

[11] Tseng, and C. Wei, "Lock your designs with the virtex-4 security solution", XCell Journal, XILINX, Spring, 2005.

[12] Xilinx, "Virtex-6 FPGA configuration user guide", `http://www.xilinx.com/support/documentation/user_guides/ug360.pdf`.

[13] A. Aziz, and N. Ikram, "An FPGA-based AES-CCM crypto core for IEEE 802.11 i Architecture", International Journal of Networks Security, volume=5, pp. 224–232, 2007.

[14] S. Drimer, T. Guneysu, and C. Paar, "DSPs, BRAMs, and a pinch of logic: Extended recipes for AES on FPGAs", ACM Transactions on Reconfigurable Technology and Systems (TRETS), vol. 3, 2010.

[15] A. Satoh, S. Morioka, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S box Optimization", Advances in Cryptology-ASIACRYPT 2001, pp. 239–254, 2001.

[16] A. Satoh, T. Sugawara, and T. Aoki, "High-speed pipelined hardware architecture for Galois counter mode", Journal of Information Security, pp. 118–129, 2007.

[17] Altera whitepaper, "Design Security in Stratix III Devices", http://www.altera.com/literature/wp/wp-01010.pdf, 2010.

[18] Xilinx commercial brochure, "Lock Your Designs with the Virtex-4 Security Solution", http://www.xilinx.com/publications/xcellonline/xcell/52/xc/pdf/xc/v4security52.pdf.

[19] Xilinx, "Virtex-5 FPGA Data Sheet: DC and Switching Characteristics", http://www.xilinx.com/support/documentation/data/sheets/ds202.pdf.

[20] Xilinx, "Spartan-3 FPGA family: Complete data sheet", http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf.

[21] Cisco Corporation, "Cisco ASA 5500 Series Adaptive Security Appliances", http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6032/ps6094/ps6120/prod-brochure0900aecd80285492.pdf, 2010.

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

[23] R. Vaslin, G. Gogniat, J. Diguet, R. Tessier, D. Unnikrishnan, and K. Gaj, "Memory Security Management for Reconfigurable Embedded Systems", International Conference on ICECE Technology, FPT, pp. 153–160, 2008.

[24] J. Wang, G. Shou, Y. Hu, and Z. Guo, "High-Speed Architectures for GHASH Based on Efficient Bit-parallel Multipliers", IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS), pp. 582–586, 2010.

[25] 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.