Microprocessors and Microsystems 45 (2016) 129–140

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

GCM implementations of Camellia-128 and SMS4 by optimizing the polynomial multiplier

Alberto F. Martínez-Herrera[a,*], Cuauhtemoc Mancillas-López[b], Carlos Mex-Perera[a]

[a] Department of Electrical and Computer Engineering, Tecnológico de Monterrey, Monterrey Campus, N. L, 64869, México
[b] Laboratoire Hubert Curien UMR CNRS 5516, Université Jean Monnet, Saint Etienne, France

ARTICLE INFO

Article history:
Received 11 April 2015
Revised 21 January 2016
Accepted 14 April 2016
Available online 16 April 2016

Keywords:
GCM
FPGAs
Block ciphers
Pipeline
Architectures

ABSTRACT

In some scenarios, the cryptographic primitives should support more than one functionality. Authenticated Encryption/Verified Decryption (AEVD) combines encryption and authentication at the same time, which is useful in communication protocols (DNS, IPSEC, etc.). Nevertheless, authenticated encryption needs some optimizations to ensure fast performance. One solution could be the use of the Galois Counter Mode (GCM) scheme. To reach fast performances, this work broadens some GCM models described in Chakraborty et al.'s [D. Chakraborty, C. Mancillas Lopez, F. Rodriguez Henriquez, P. Sarkar, Efficient hardware implementations of BRW polynomials and tweakable enciphering schemes, Comput IEEE Trans 62 (2) (2013) 279–294, doi:10.1109/TC.2011.227] work with two changes. The first one is focused on speeding-up the polynomial multiplier necessary to perform the authentication process. That polynomial multiplier is extended for supporting four stages, based on the well-known Karatsuba–Ofman algorithm. The second one is the modification of two known block ciphers such as Camellia-128 and SMS4 with the GCM scheme. The constructed GCM is able to support variable-length messages greater than 512 bits. The throughput of the polynomial multiplier is greater than 28 Gbps for all the tested platforms. The independent block ciphers in encryption-only mode reach a throughput greater than 28 Gbps, and for all the GCM cases reported in this manuscript the throughput is greater than 9.5 Gbps.

1. Introduction

Several applications need to support authentication and integrity tasks. Mainly, those which need to ensure that a message has been received without modifications during the communication process. Public key cryptography might be a solution, but due to its computation cost it cannot be useful. In real application, public key cryptography is complemented with fast symmetric cryptographic primitives as we can find in the Socket Security Layer (SSL) protocol, where a stage based on public key cryptography establishes a symmetric key session and authentication of the pairs using digital certificates. After that, symmetric encryption is used to encrypt and authenticate the communication until the SSL session finishes. Then, it is necessary to develop new strategies to ensure the integrity and privacy of a message depending on a given scenario. One of them is the proposed by McGrew and Viega, denoted as Galois Counter Mode operation (GCM) [17], which offers this two services. In addition to that properties, the GCM is aimed at scenarios where speed is the main constraint to be optimized. Some examples can be found in communication systems like wired and wireless networks, optic fiber, digital TV, etc.

The main feature that the GCM schemes hold is the use of universal hashing constructed with polynomial multipliers. Their use is associated to some desirable properties: scalability and versatility. The former refers that the use of a polynomial multiplier does not increase the complexity of the complete cryptographic primitive/algorithm related to efficiency and performance. The latter means that a polynomial multiplier can be arranged into different architectures either sequential, parallel or pipeline. Additionally, the use of a key enhances the hashing properties of this scheme.

The content of this manuscript is divided as follows. Section 2 lists the different AEVD schemes. Section 3 describes the mathematical operations involved in Camellia-128 and SMS4. Section 4 refers to the typical GCM schemes. Section 5 lists works regarding Camellia-128 and SMS4 into GCM schemes and additional examples with the Advanced Encryption Standard (AES). Section 6 describes the pipeline versions of Camellia-128, SMS4 and their inclusion into the GCM scheme. Section 7 describes the

[*] Corresponding author. Tel.: +527717161647.
E-mail addresses: a00798620@itesm.mx, alberto.herrera.tec@gmail.com (A.F. Martínez-Herrera), cuauhtemoc.mancillas.lopez@univ-st-etienne.fr (C. Mancillas-López), carlosmex@itesm.mx (C. Mex-Perera).

http://dx.doi.org/10.1016/j.micpro.2016.04.006
0141-9331/© 2016 Elsevier B.V. All rights reserved.

results of the GCM schemes and Section 8 summarizes the conclusions of this work.

1.1. Notation

Let X and Y be binary strings, $X||Y$ is their concatenation and $|X|$ is the length of X in bits. $A = A_1||A_2||...||A_m$, $P = P_1||P_2||...||P_t$ and $C = C_1||C_2||...||C_t$ are associated data, plaintext and ciphertext respectively. $|A_j| = |P_i| = |C_i| = n$ for $1 \le j < m$, $1 \le i < t$. For last blocks $|A_m| \le n$ and $|P_t| = |C_t| \le n$, they can also be treated as polynomials in $GF(2^n)$. $E_K(X)$ is the encryption of X using the underlying block cipher with key K, and the block size is n. $\mathrm{MSB}_u(X)$ refers to the u most significant bits of X. $\{x\}^q$ refers to a sequence of q bits, where $x \in \{0, 1\}$. In this work $n = 128$.

2. Authenticated encryption/decryption schemes, background and taxonomy

Block ciphers can encrypt/decrypt $n - bit$ messages where n is called its block-size. For real applications, messages are much greater than n bits. In this case the use of an mode of operation is necessary. The classical modes of operation of block ciphers are listed below:

- Electronic code book (ECB). Each block of 128 bits is encrypted/decrypted independently of the rest as $ECB_K(P_1 \parallel ... \parallel P_t) = E_K(P_1) \parallel ... \parallel E_K(P_t)$. This mode is not recommendable since the encryption of equal blocks generates the same ciphertext $E_K(P_1) = E_K(P_2)$ when $P_1 = P_2$, this allows a widely known *chosen plaintext attack*.
- Cipher block chaining (CBC). This mode uses an initial vector IV and encrypts in the following form $CBC_K^{IV}(P_1 \parallel ... \parallel P_t) = C_1 \parallel ... \parallel C_t = E_K(IV \oplus P_1) \parallel E_K(C_1 \oplus P_2) \parallel ... \parallel E_K(C_{t-1} \oplus P_t)$.
- Output feedback (OFB). First of all a binary string is obtained using the IV as $R_1 \parallel ... \parallel R_t = E_K(IV) \parallel E_K(R_1) \parallel ... \parallel E_K(R_{t-1})$ and then $OFB_K^{IV}(P_1 \parallel ... \parallel P_t) = R_1 \oplus P_1 \parallel ... \parallel R_t \oplus P_t$.
- Cipher feedback (CFB). Regarding the OFB mode, this mode uses the encryption of the previous block as input of the next block cipher, $CFB_K^{IV}(P_1 \parallel ... \parallel P_t) = C_1 \parallel ... \parallel C_t = E_K(C_0) \oplus P_1 \parallel E_K(C_1) \oplus P_2 \parallel ... \parallel E_K(C_{t-1}) \oplus P_t$, where $C_0 = E_K(IV)$.
- Counter (CTR). An IV is used as starter value of a counter and the encryption is performed as $Ctr_K^{IV}(P_1 \parallel ... \parallel P_t) = C_1 \parallel ... \parallel C_t = E_K((IV + 1)) \oplus P_1 \parallel E_K((IV + 2)) \oplus P_2 \parallel ... \parallel E_K((IV + t)) \oplus P_t$. It is recommended to use as IV the concatenation of counter and a nonce (a value unique per each message).

In the above definitions the decryption equations were omitted, as they can be easily deducted from the encryption equations. Classical modes of operation offer only privacy, i.e., they protect the information from unauthorized access.

Authenticity and integrity of messages are also important security services. Data integrity warranties that if a message is corrupted, this change can be detected. Here, a key may not be necessary, while data authenticity involves the use of a key. Nevertheless, both properties can be achieved by using Message Authentication Codes (MACs), which are keyed algorithms. A MAC algorithm outputs a tag T for authentication which depends of all the message as follows $T = MAC_K(P)$. During the communication, the message transmitted is $P||T$ and when the message is received one can verify its authenticity by $T' = MAC_K(P)$. If $T = T'$ then the message is valid, otherwise the message was corrupted. Some MACs also used a *Nonce* that is a unique parameter for each message. MAC algorithms have similar properties to cryptographic hash functions. In fact, MACs can be constructed using cryptographic hash functions as the widely known HMAC [15].

CBC-MAC is the MAC algorithm using CBC mode of operation with $IV = 0$ and the block C_t is the tag because it depends of the complete message. CBC-MAC has many security problems [18]. In [8], CBC-MAC is improved and three new MAC algorithms based on it using three keys were presented: ECBC, FCBC, and XCBC. Iwata and Kurosawa present a version of CBC-MAC using one key in [12] and then its construction was taken as an standard [10]. Other way to construct MACs is using polynomial hashes in combination with block ciphers, for instance Poly1305 [[6]]. In general, these algorithms are sequential, PMAC [7] allows to be implemented either in parallel or pipeline.

There are practical scenarios where some data needs to be enciphered, while the rest only needs to be authenticated. For instance, in the Domain Name System (DNS) protocol, there are four fields in the packet that form the DNS datagram: header, answer, authority and additional. While the first one indicates how the packet will be processed, the rest of the fields contain information about the servers that were consulted to reach a successful query, including the desired answer. With the current surveillance scenarios promoted by some rogue entities, users wish to encrypt data that they consider sensitive in terms of eavesdrop. From the user perspective, the last three fields, that contain sensitive information, may be encrypted. Naturally, to avoid some malfunction of the DNS protocol, the header should be treated as the meta-data of the DNS packet and should only be authenticated. Of course, additional measures of key management should be solved but that problem can be overcome with the use of public key cryptography, leaving to the AEVD the use of a session key (and additional parameters) to perform the corresponding AEVD operations. A DNS packet can be embedded into an Ethernet/UDP/IP packet, where each one has its own structure, it is necessary to distinguish within such structures the information that must be authenticated (packet headers) from the data to be encrypted. Besides, the header length can be expanded when the DNS packet is processed through the protocol stack.

Aside the GCM, the most common AEVD schemes are listed below.

- Counter with CBC-MAC (CCM). It was established as a standard by the NIST [10]. It combines the CBC-MAC to compute T and Counter mode for encryption. Here, T is also encrypted. Apart of its original appearance, the IETF adopted this encryption-authentication mode to be applied on different environments such as IPsec as described by the RFCs 3610 and 4309 [11,31].
- Offset codebook mode (OCB). Proposed by Rogaway et al. [24], this mode is based on the ECB mode of operation, but enforced with XOR operations using a different value per each block before and after the cipher call. OCB allows implementations either in parallel and pipeline.
- Carter Wegman counter (CWC). Based on the universal hash functions studied by Wegman and Carter [29]. This mode is in some way one of the first references previous to the creation of GCM. It was one of the first modes to treat the encryption and the authentication mode together. Thus, this mode can be easily configured into different architectures: pipeline and parallel. Its drawback is that it uses integer multiplications [14].

3. GCM scheme

The general way to implement GCM scheme is shown in Fig. 1, given by McGrew and Viega [17]. We know that P and C are variable-length so that both should be divided to be processed, and T is the tag. As established in the GCM standard documentation, the parameter H should be obtained as $E_K(\{0\}^{128})$. Y_0 is obtained from the input IV and then it is increased each time as Y_0, Y_1, ..., Y_t, where t is the number of 128-bit blocks in P. Each Y_i is then