FPGA-Based High-Speed Authenticated Encryption System

Michael Muehlberghuber, Christoph Keller, Frank K. Gürkaynak, Norbert Felber

FPGA-Based High-Speed Authenticated Encryption System

Michael Muehlberghuber, Christoph Keller,
Frank K. Gürkaynak, and Norbert Felber

Integrated Systems Laboratory (IIS), ETH Zurich,
Gloriastrasse 35, 8092 Zurich, Switzerland
{mbgh,chrikell,kgf,felber}@iis.ee.ethz.ch

Abstract. The Advanced Encryption Standard (AES) running in the Galois/Counter Mode of Operation represents a de facto standard in the field of hardware-accelerated, block-cipher-based high-speed authenticated encryption (AE) systems. We propose hardware architectures supporting the Ethernet standard IEEE 802.3ba utilizing different cryptographic primitives suitable for AE applications. Our main design goal was to achieve high throughput on FPGA platforms. Compared to previous works aiming at data rates beyond 100 Gbit/s, our design makes use of an alternative block cipher and an alternative mode of operation, namely Serpent and the offset codebook mode of operation, respectively. Using four cipher cores for the encryption part of the AE architecture, we achieve a throughput of 141 Gbit/s on an Altera Stratix IV FPGA. The design requires 39 kALMs and runs at a maximum clock frequency of 275 MHz. This represents, to the best of our knowledge, the fastest full implementation of an AE scheme on FPGAs to date. In order to make the design applicable in a real-world environment, we developed a custom-designed printed circuit board for the Stratix IV FPGA, suitable to process data with up to 100 Gbit/s.

Keywords: Authenticated encryption, High-throughput architecture, FPGA, Pipelining, Serpent, OCB, AES, GCM.

1 Introduction

Confidentiality and authenticity are two of the most important cryptographic goals. Whereas the former assures that any eavesdropping adversary is unable to decipher a given message—even if she has access to the transmission medium—, the latter refers to the cryptographic service that ensures that the receiver of a message can be sure about its origin, i.e., that an attacker has not impersonated the sender. Authenticated encryption (AE) combines these two services and allows a *secure* and *authentic* communication between two parties.

In order to provide high-throughput AE implementations based on block ciphers, so-called *combined* modes of operation have been designed throughout the last decade. They allow a higher throughput by interleaving the authentication part and the encryption part instead of calculating them consecutively

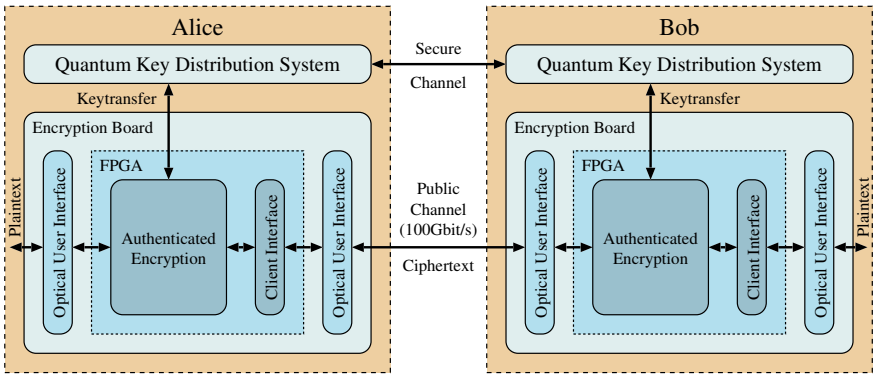

Fig. 1. High-speed authenticated encryption system setup

(as traditional AE methods do). The two most widely accepted AE modes of operation are *Counter with CBC-MAC (CCM)* [19] and *Galois/Counter Mode (GCM)* [11]. Their acceptance is most likely due to the fact that they have been recommended by the National Institute of Standards and Technology (NIST) (cf. [5] and [6]). Since then, they have been applied to technologies and protocols such as WiFi 802.11 [8] and IPsec [17]. Although the specifications of these modes do not determine the underlying block cipher, most applications make use of the Advanced Encryption Standard (AES) [14] since it is another algorithm standardized by the NIST.

The present work proposes a block cipher-based hardware architecture for AE, targeting high throughput on field-programmable gate array (FPGA) platforms. Our design has been developed as to fulfill the requirements of the Ethernet standard IEEE 802.3ba [1], which allows for transmission speeds of up to 100 Gbit/s. This work has been designed as part of a system that employs quantum key distribution (QKD) for synchronizing multiple private key exchanges within a single second, and provides authenticated encryption service using conventional cryptographic primitives. Fig. 1 illustrates the overall system setup. The main contributions of our work are related to the *Authenticated Encryption* part of Fig. 1, i.e., the digital, AE-related parts on the FPGA and have originally been presented in [12].

So far, our system employed a common GCM-AES-based cryptographic primitive in order to achieve the required throughput. In this work, we examine alternatives for both the block cipher and the mode of operation and compare the performance of these alternatives to the established cryptographic primitives. Besides exploring more efficient hardware implementations, this work is also motivated by providing an alternative AE scheme, in case successful attacks are developed against the existing primitives. We evaluate the Serpent block cipher [3] and the offset codebook (OCB) mode of operation [16] and we provide results of hardware implementations for different mode of operation/block cipher combinations, namely:

- OCB-Serpent
- OCB-AES
- GCM-Serpent
- GCM-AES

Our fastest AE implementation is based on an OCB-Serpent architecture and requires 39 kALMs (Adaptive Logic Modules) on an Altera Stratix IV FPGA. It uses four cipher cores for the encryption part and reaches a throughput of 141 Gbit/s, running at 275 MHz.

Moreover, we developed a custom-designed printed circuit board (PCB), which allows us to use the presented designs in real-world applications such as the system illustrated in Fig. 1. So far, two copies of the board have been fabricated and successfully tested in various sample experiments.

The remainder of this work is structured as follows. In the next section, we present an overview of related work on hardware architectures targeting high-throughput AE designs. In Section 3, a description of Serpent and OCB is given. The actual hardware architecture of our design is presented in Section 4. Throughout Section 5, we summarize our results, including a brief discussion. Finally, Section 6 provides a description of the custom-designed PCB including some of its major features, before we conclude our work in Section 7.

2 Related Work

Due to the standardization by the NIST, GCM-AES has received significant attention from both the research community and the industry, and several implementations targeting FPGAs can already be found in the literature.

In 2009, Zhou et al. [20] presented a single-core GCM-AES design, which targets a Xilinx Virtex-5 FPGA. They achieved a throughput of 41.5 Gbit/s using the 128-bit version of AES. Henzen and Fichtner [7] showed that it is possible to break the 100 Gbit/s barrier on a Virtex-5 FPGA platform. They made use of four fully unrolled AES cores for the encryption part and used four Karatsuba-Ofman (KO) multipliers in order to realize the authentication part. Their design reaches a throughput of 119.3 Gbit/s.

The most complex operation during the computation of a message digest according to GCM is the multiplication in the binary finite-field $GF(2^{128})$, which is part of the universal hashing function called GHASH. Therefore, most of the effort in improving GCM implementations has been spent on speeding up this calculation. Wang et al. [18] presented a GHASH architecture based on four GHASH cores that achieved a throughput of 123.1 Gbit/s on a Virtex-5. Crenne et al. [4] reached 238.1 Gbit/s by using 8 parallel finite-field multipliers, also targeting a Xilinx Virtex-5 FPGA. Since we aim at a full AE architecture, i.e., a design including both the authenticity and the confidentiality part, we do not consider these GHASH-only implementations for our investigations.

To the best of our knowledge, no hardware architecture based on a block cipher other than AES and targeting a high-throughput AE implementation has been presented so far. Moreover, no AES design has been published to date, which makes use of an operation mode different than GCM in order to achieve throughputs up to 100 Gbit/s.

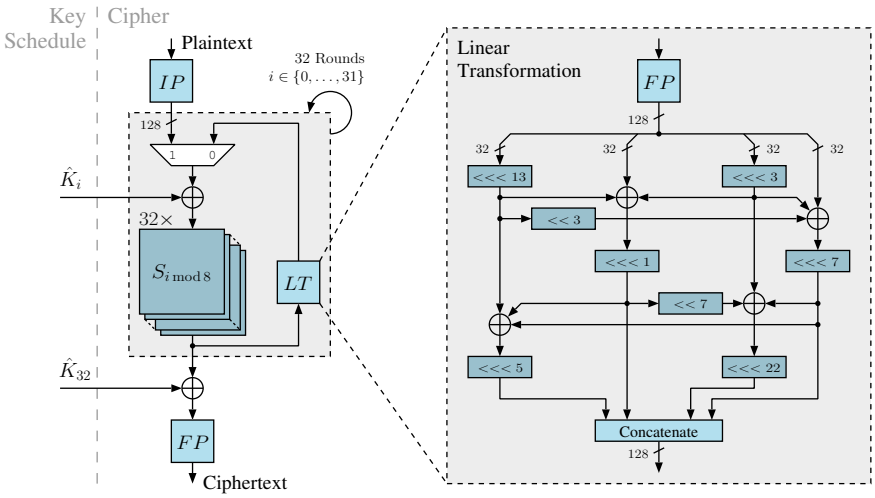

Fig. 2. Serpent block cipher

3 100 Gbit/s Authenticated Encryption Alternatives

In order to reach throughputs exceeding 100 Gbit/s on commercial FPGA devices, it is necessary to use multiple parallel instances of cryptographic primitives. Although AES running in GCM mode is currently the most widespread option for high-throughput hardware architectures, using these cryptographic primitives is not a requirement. Different block ciphers and modes of operation, like the ones presented in the following sections, can be used for a throughput-oriented AE system as well.

3.1 Serpent Block Cipher

Serpent was the runner-up of the AES block cipher competition. Although it has not been chosen by the NIST during the competition, it was considered to be a close alternative and is still known to be secure from a cryptographic point of view as the considerably large number of rounds contributes to its security [13]. In the following we will briefly discuss the main components of Serpent, i.e., the key schedule and the cipher itself, using the *conventional* implementation described in the official proposal [2]. In order to change from the *conventional* to the *bitslice* version of Serpent[1], all instances of the initial and the final permutation have to be omitted.

Cipher. Fig. 2 illustrates the Serpent cipher which consists of an initial permutation (IP), 32 round transformations, and a final permutation (FP). The

[1] We refer to the Serpent proposal [2] for further information on the *bitslice* implementation.

first 31 rounds of the cipher include a *key-mixing stage*, a *substitution stage*, and an *avalanche stage* (i.e., a stage where a linear transformation takes place). In the last round of the cipher, the linear transformation is omitted and replaced by another key-mixing operation. Serpent makes use of eight different S-boxes ($S_i, i \in \{0 \dots 7\}$) which repeat themselves every eighth round as shown in Fig. 2. Note that only a single S-box is used within each round of the cipher.

Key Schedule. The key schedule of Serpent takes a 256-bit cipher key K and expands it to thirty-three 128-bit subkeys denoted by $\hat{K}_i$. Cipher keys shorter than 256 bits are padded by appending a single "1", followed by as many "0"s as required in order to reach a length of 256 bits. After padding, K gets expressed using eight 32-bit values, i.e., $K = \{\omega_{-8}, \dots, \omega_{-1}\}$, and extended to an intermediate key $\{\omega_0, \dots, \omega_{131}\}$ according to

$$\omega_i = (\omega_{i-8} \oplus \omega_{i-5} \oplus \omega_{i-3} \oplus \omega_{i-1} \oplus \phi \oplus i) <<< 11\,, \qquad i \in \{0 \dots 131\}\,,$$

where $<<< i$ denotes a rotate-left function by i bits and $\phi = 0x9E3779B9$, i.e., the 32-bit value of the fractional part of the golden ratio. The actual subkeys, which are required during the round transformations of the cipher, are finally obtained by

$$\hat{K}_i = IP(S_{(3-i) \bmod 8}(\omega_{4i}, \omega_{4i+1}, \omega_{4i+2}, \omega_{4i+3}))\,, \qquad i \in \{0 \dots 32\}\,,$$

where S_i refers to one of the eight Serpent S-boxes. Similar to the cipher, the *bitslice* implementation of the Serpent key schedule can be obtained by removing all instances of the initial permutation IP. For a detailed description of Serpent, including additional information about the initial and the final permutation, we refer the reader to the official Serpent proposal [2].

3.2 Offset CodeBook Mode

The offset codebook (OCB) block cipher mode of operation is a combined AE scheme and has first been published by Rogaway et al. [16] in 2001. It is strongly related to the Integrity Aware Parallelizable Mode (IAPM) by C. Jutla [9] and three different versions have been made public since 2001. Throughout the remainder of this work we solely refer to the third version of it, i.e., OCB3 [10].

To start the authenticated encryption scheme according to OCB, a plaintext message, denoted by M, gets split into m different blocks, each of length n and an optional block M_* of length smaller than n as follows[2]:

$$M = \begin{cases} M_1, \dots, M_m\,, & \text{if } |M| = k \cdot n \text{ and } k \in \mathbb{N}\,, \\ M_1, \dots, M_m, M_*\,, & \text{else}\,. \end{cases}$$

Algorithm 1 and Fig. 3 describe the authenticated encryption according to OCB using pseudo-code and a block diagram, respectively. For simplicity, only

[2] We refer to the length of x in bits using the following notation: $|x|$

Algorithm 1 OCB authenticated encryption.

Input: Message M, Message block length n, Cipher key K, Nonce N, Associated data A, Authentication tag length τ, $0 \leq \tau \leq 128$
Output: Ciphertext C, Authentication tag T

1: **if** $|N| \geq n$ **then return** INVALID
2: $\{M_1, \ldots, M_m, M_*\} \leftarrow M$, with $|M_i| = n$ and $|M_*| < n$
3: $Checksum \leftarrow 0^{128}; C \leftarrow 0^{128}$
4: $L_*, L_\$, L[0] \ldots L[\lfloor log_2(m) \rfloor] \leftarrow Setup(K, m)$
5: $\Delta \leftarrow Init(N, n, K)$
6: **for** $i = 1$ to m **do**
7: $\quad \Delta \leftarrow \Delta \oplus L[ntz(i)]$ ▷ $Inc_i(\Delta)$
8: $\quad C \leftarrow C || E_K(M_i \oplus \Delta) \oplus \Delta$
9: $\quad Checksum \leftarrow Checksum \oplus M_i$
10: **end for**
11: **if** $M_* \neq \varnothing$ **then**
12: $\quad \Delta \leftarrow \Delta \oplus L_*$ ▷ $Inc_*(\Delta)$
13: $\quad Pad \leftarrow E_K(\Delta)$
14: $\quad C \leftarrow C || M_* \oplus (Pad \wedge (1^{|M_*|}))$
15: $\quad Checksum \leftarrow Checksum \oplus M_* 10^*$, with
$\qquad M_* 10^* = M_* \| 1 \| 0 \ldots 0$, such that $|M_* 10^*| = n$
16: **end if**
17: $\Delta \leftarrow \Delta \oplus L_\$$ ▷ $Inc_\$(\Delta)$
18: $Final \leftarrow E_K(Checksum \oplus \Delta)$
19: $Auth \leftarrow Hash_K(A)$
20: $Tag \leftarrow Final \oplus Auth$
21: $T \leftarrow trunc(Tag, \tau)$
22: **return** $C || T$

the cases for full message blocks, i.e., $M_* = \varnothing$ is shown in Fig. 3. The cipher starts with a setup and initialization step (cf. line 4 and 5 in Algorithm 1). Thereafter, each message block can be processed independently of each other (line 6 to 16). Finally, the authentication tag T is determined throughout line 17 to 21. The characters $||$, $\oplus$, and $\wedge$ represent the concatenation, bitwise exclusive or, and bitwise and operation. The term *ntz(i)* describes the number of trailing zeroes of i in binary representation. 0^n and 1^n stand for bit strings of length n containing only zeros and ones, respectively. Furthermore, $trunc(X, y)$ truncates the bit string X to its y least significant bits. We use $\varnothing$ to represent an *empty set*. Appendix A provides listings for the *Setup*, *Init*, and $Hash_K$ procedures used throughout Algorithm 1.

When using a counter for the nonce N, the calculation of the initial offset Δ requires a block cipher call only every $64th$ initialization. This is due to the fact that the least significant six bits of N are set to zero before passing it to the block cipher (cf. line 3 of Algorithm 3). This fact together with the parallelizable processing of the message blocks, makes OCB suitable for high-throughput applications.

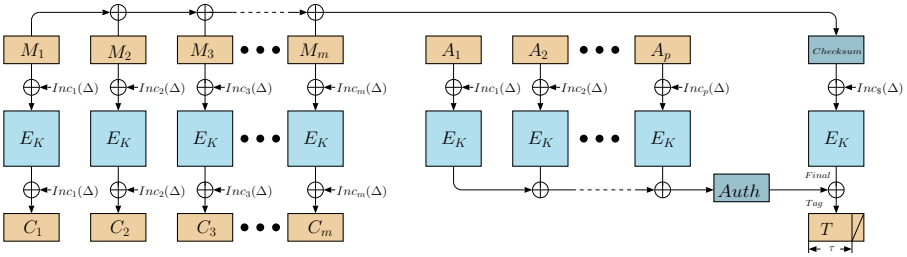

Fig. 3. Overview of the encryption and the authentication part of OCB

4 OCB-Serpent Hardware Architecture

For the OCB-Serpent architecture, supporting the IEEE 802.3ba Ethernet standard, we assume the following prerequisites:

- The size of the message block counter i is restricted to 7 bits, since 2^7 message blocks are capable of hosting a full Ethernet frame.
- As our target application ensures solely full message blocks, we do not handle short final message blocks separately.

As previously mentioned, to achieve extremely high throughputs, a multi-core approach has proven to be the only viable option when implementing AE on commercial FPGA platforms. Similar to GCM, OCB also allows two successive message blocks to be processed independently of each other. We have taken advantage of this fact and decided to use four parallel cipher cores in order to achieve the desired throughput. Fig. 4 illustrates the OCB architecture for authenticated encryption based on four Serpent cores.

4.1 Pipelined Four-Core Serpent Architecture

Each of the four Serpent cores handles a single 128-bit message block. Therefore, the overall design can process a 512-bit message at a time. As can be seen from Fig. 4, we fully unrolled the 32 rounds of Serpent. Furthermore, we inserted pipeline stages after each round in order to increase the maximum frequency of the cipher cores. Although the pipelined architecture allows us to clock the Serpent cores at a higher frequency, one problem inherent to all pipeline architectures has to be taken into consideration: When the normal flow of operations has to be suspended, the entire pipeline must be stopped in order to allow the rest of the operation to resume. Such an occurrence is known as a *pipeline stall* and can, for instance, occur during a key change.

Due to the unrolling of the Serpent rounds, we have to realize 1024 of the 4-bit S-boxes for each core, which requires a considerable amount of resources. The subkeys for the key-mixing stage of all four Serpent cores are provided from a single key schedule, i.e., the cores always operate on the same cipher key.

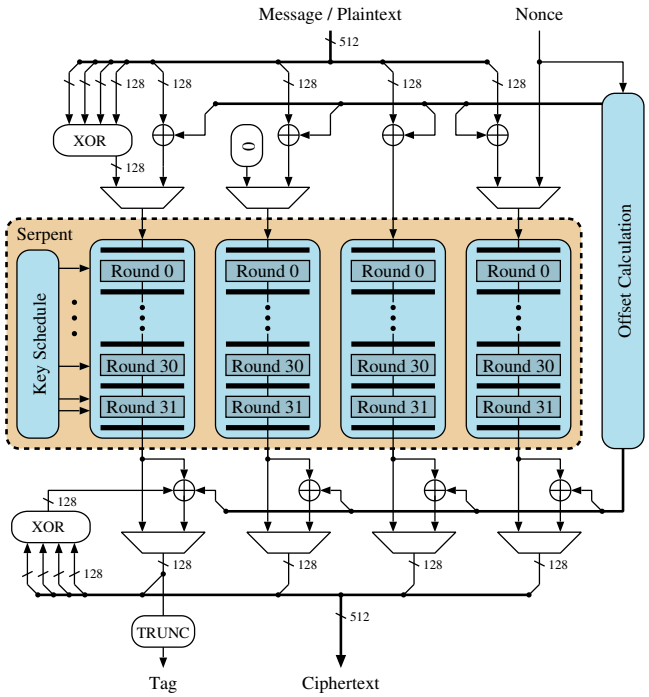

Fig. 4. OCB-Serpent architecture

4.2 OCB - Authenticated Encryption

OCB can, in general, be subdivided into three stages: *Initialization*, *encryption/authentication*, and *finalization*. During the initialization phase, two potential pipeline stalls may occur if not handled properly. First, after each key change a cipher call $E_K(0^{128})$ is required in order to compute the table values $L[..]$ (cf. Algorithm 2). Second, each new message needs a fresh nonce N, and thus a new offset value Δ. Since the calculation of the initial offset also requires a cipher call, this may result in another pipeline stall. In order to reduce the number of pipeline stalls to a minimum, we precompute the initial offset values.

The limitation of message lengths to a maximum of 2^7 blocks facilitates the precomputation of the $L[..]$-values, as it limits the maximum number of trailing zeroes of the block counter i to six. Thus, only $L_\$$, L_*, and $L_0 \dots L_6$ have to be precomputed and stored in registers. In fact, when the result of $E_K(0^{128})$ is available, all nine table values can be computed in a single clock cycle[3].

When processing a block in authenticated-encryption mode, the message gets XOR-ed with the current offset Δ_i, encrypted, and finally XOR-ed with Δ_i again. As pipeline stages were introduced into the cipher cores, the Δ-values either have

[3] The calculation only depends on operations cheap to implement in hardware, i.e., fixed shift and conditional exclusive or operations.

to be stored or recalculated. We decided to recompute the offsets as it makes the implementation less dependent on the underlying block cipher and the number of pipeline stages used. Since the multi-core design processes four message blocks in parallel, the offset-calculation needs to be capable of providing four offset values at a time. In fact, the calculation of the four offset values is relatively cheap as the initial offset only has to be XOR-ed with the precomputed table values.

As described in Algorithm 3, a nonce-dependent call to the encryption of the block cipher is required. The result of this operation, further-on called *Ktop*, then has to be shifted by a 6-bit nonce-dependent value *Bottom*. First, in order to be able to perform this shift-operation, *Bottom* has to be buffered until the result of $E_K(Top)$ is available. Second, the 6-bit variable shift is done using a 192-bit by 6-bit barrel shifter. Although using a counter for the nonce N could avoid the resource-expensive barrel shifter, we decided to keep it in order to stay independent of the actual structure of the chosen nonce.

4.3 Decryption

Authenticated decryption according to OCB is very similar to the encryption process. Exchanging the encryption operation E_K of the underlying cipher by the decryption operation D_K in line 8 of Algorithm 1, turns OCB into decryption mode. However, the other encryption operations remain. Therefore, the multi-core decryption unit contains four block-cipher decryption cores, one encryption core, and a common key schedule. In order to assure authenticity of a provided message, the re-calculated message tag T' must be equal to the tag T, received from the opposite communicating party.

A minor drawback of authenticated decryption according to OCB is the fact that a delay, dependent on the number of pipeline stages p between the calculation of the plaintext and the calculation of the authentication tag exists. This delay is caused by the calculation of the authentication tag which requires the *Checksum* of the plaintext. Therefore, in order to verify the authentication tag of a message, p plaintext blocks have to be buffered, resulting in additional memory requirements.

5 Results

We coded our architectures in VHDL. For the synthesis and place&route design steps, we used Altera Quartus II version 11.0. Functional correctness was verified with Modelsim 6.6e simulator. Synthesis was conducted using a speed-optimzed setting. Except of M9K block memories, no Altera-specific logic blocks had been used. In order to have some reference implementations regarding AES and GCM on our target platform (Altera Stratix IV), we also synthesized a four-core AES cipher architecture as well as GCM based on both Serpent and AES. The AES architectures were accomplished with an underlying four-core AES similar to the one proposed by Henzen et al. [7]. We used the 128-bit version of AES and

Table 1. Encryption-only and authenticated encryption results targeting an Altera Stratix IV (EP4S100G5F45) platform using four cipher cores

Block Cipher	Mode of Operation	Area [ALMs]	Area [M9K Bl.]	f_{max} [MHz]	Throughput [Gbit/s]	Throughput [%]
Cipher-Only Architectures						
Serpent	cipher-only	28,399	0	281	144	136
AES	cipher-only	7,661	314	267	137	130
Authenticated Encryption Architectures						
Serpent	OCB	38,312	0	275	141	133
AES	OCB	11,948	314	250	128	121
Serpent	GCM	56,474	0	203	104	99
AES	GCM	24,313	314	206	105	100

fully unrolled the 10 rounds. Furthermore, pipelining registers have been inserted after each round similar to the Serpent cipher core design.

The first two rows of Table 1 contain the place&route results of the multi-core encryption architectures (i.e., without running any mode of operation). The subsequent rows present the results for the different combinations of block ciphers and modes of operation. Regarding the block ciphers, the fully unrolled four-core AES design requires less area as it only has to provide 160 8-bit S-boxes for each cipher core, compared to the 1024 4-bit S-boxes needed by the Serpent cores. For the AES cores we utilized the M9K memory blocks in order to implement the 160 S-boxes, whereas for Serpent we implemented them solely in look-up tables. One of the reasons for this design decision was the significant routing overhead, which would have been required for the 1024 Serpent S-boxes being realized in M9K memory blocks. Since the high-throughput universal hash function GHASH of the GCM mode occupies a lot of area, OCB designs result in a smaller footprint than their GCM counterparts.

Regarding the throughput, all architectures met the target of 100 Gbit/s. However, the OCB versions are considerably faster. This is mainly because of the simpler architecture of OCB compared to GCM, which requires the resource-expensive GHASH function. Table 1 shows that the critical path of the GCM-based architectures is dominated by the authentication part, whereas the OCB-based designs almost reach the maximum frequency of the cipher-only implementations (with minor exceptions which are most likely due to placement and routing disparities). Compared to our results in [12], we were able to further increase the maximum frequency of both our cipher-only architectures as well as the designs based on OCB. Although our results show that OCB is at least as suitable for high-throughput hardware implementations as GCM, the latter is still the preferred mode of operation in the literature when designing high-speed authenticated encryption hardware architectures based on block ciphers. This might be due to the facts that there are some US patents on OCB and that,

Table 2. Block cipher modes of operation comparison

Property	**OCB**	**GCM**
Patented	Yes	No
Parallelizable	Yes (Encr. + Auth.)	Yes (Encr. + Auth.)
Decryption required	Yes	No
Cipher calls (Initialization)	1	1
Cipher calls (Encryption)	$\lceil \lvert M \rvert / n \rceil + 1.016$[1]	$\lceil \lvert M \rvert / n \rceil + 1$

[1] Applies as long as the associated data is fixed during a single session and a counter is used for the nonce.

in contrast to GCM, it has not been recommended by the NIST. Regarding the patents on OCB, its author has recently eased licensing for a variety of applications [15] what may increase the popularity of OCB in the near future. Table 2 summarizes the properties of the two AE block cipher modes of operation. One benefit of GMC might be that it solely requires the encryption of the underlying block cipher whereas OCB also needs the decryption.

Our designs have been tested on a self-designed printed circuit board (PCB), which has solely been developed for high-speed authenticated encryption architectures running on an Altera Stratix IV FPGA.

6 100 Gbit/s Authenticated Encryption System Design

The AE core, described in Section 4, was developed as part of a larger FPGA-based system that will be used to encrypt data on IEEE 802.3ba Ethernet connections, allowing data rates of up to 100 Gbit/s. Designing a real system that is able to reliably process such high data-throughputs poses some formidable challenges. We have successfully developed a complete FPGA-based system working at 40 respectively 100 Gbit/s and will present both the digital part and the PCB development throughout the next sections.

6.1 FPGA Digital Design

Processing data at a rate of up to 100 Gbit/s by itself is per se rather challenging. However, transporting this amount of data to and from the processing cores is also a significant problem. In this system we have decided to aggregate plaintext data from ten separate 10 Gbit/s Ethernet links into a single 100 Gbit/s data stream. This data is then encrypted and an authentication tag is determined using the AE schemes mentioned earlier. The resulting ciphertext data stream is then transmitted over a single 100 Gbit/s Ethernet link. As illustrated in Fig. 5, the receiving path of the system works similarly in the opposite direction. The system ensures that the AE remains transparent for all 10 Gbit/s clients connected to the system.

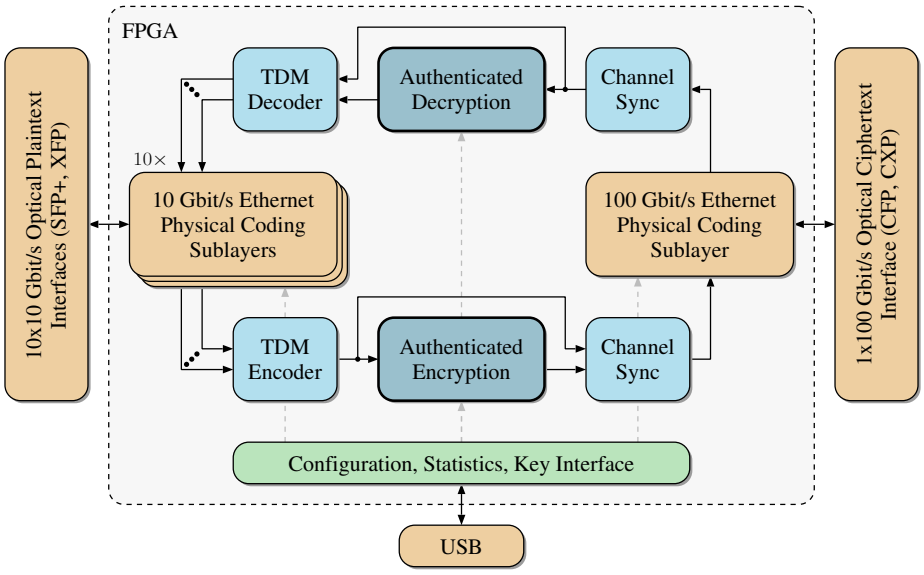

Fig. 5. Block diagram of the whole encryption system on the FPGA

Transmitting Path. The plaintext, received from the 10 Gbit/s Ethernet clients, arrives as a serial data stream. In the 10 Gbit/s Ethernet *Physical Coding Sublayer* (PCS) block, this data is parallelized and prepared for further operations. Note that in our implementation we do not require a Media Access Control (MAC) unit. Instead, we directly aggregate and encrypt the received data stream. The *TDM Encoder* collects the data from each PCS and transfers it to the *Authenticated Encryption* core, where the data is encrypted and a corresponding authentication tag is generated. The *Channel Sync* unit encapsulates the encrypted data and its authentication tag into a TDM Ethernet frame. To ensure fast resynchronization after a connection or packet loss, a synchronization frame is inserted every millisecond into the 100 Gbit/s stream. The synchronization frame is also used to transmit parameters such as the current initialization vector for GCM and the nonce for OCB. The generated frames from the *Channel Sync* block are prepared for transmission and serialized in the 100 Gbit/s PCS. The PCS uses ten 10 Gbit/s serial data streams to transmit the data.

One problem during transmission is the synchronization loss due to various effects such as electrical and optical multiplexing in the optical CFP modules and small differences in the length of electrical traces on the PCB. As a result, the serial streams may arrive at the receiver out of order. Therefore, a mechanism is required to reorder and realign the serial streams. Unique alignment markers for each stream are inserted every 100 µs to enable synchronization at the receiver.

Receiving Path. On the receiving path, the PCS deserializes the incoming 100 Gbit/s Ethernet transmission into ten 10 Gbit/s data streams. These data

streams are then reordered and possible delays are compensated by utilizing the alignment markers inserted during the transmission. The system is capable of compensating up to 200 ns of delay in this way. In the next step, the now parallelized 100 Gbit/s data stream is decoded by the receiving *Channel Sync*. If a synchronization frame is detected, parameters for the AE cores are extracted from this frame and applied for the following data frames. When a TDM Ethernet frame is detected, the payload is extracted and sent to the *Authenticated Decryption* core. In the *TDM Decoder*, the decrypted data stream is distributed to the corresponding 10 Gbit/s PCS units. In addition, the calculated authentication tag from the *Authenticated Decryption* block is compared to the received one. If an authentication failure is detected, an alert flag is set. The system can be configured to react with further measures, such as purging its input data for that channel.

An important problem of the system is clock synchronization. It occurs if the clock on the receiving side of the system differs (slightly) from the transmitting side. According to the Ethernet standard, the maximum allowed clock mismatch is 100 ppm. To be able to compensate these mismatches, the system can enlarge or shrink the gap between two Ethernet frames.

System Configuration. The system on the FPGA can be configured and monitored via a *USB* connection in the development board. Individual 10 Gbit/s links can be disabled, the encryption can be turned on or off, and secret key sizes can be determined through this interface. In addition, the encryption keys are also submitted via this interface. Moreover, the same connection can be used to monitor the operation. Statistical data such as number of transmitted and received frames, status of the *Authenticated En/Decryption* blocks, or presence and link activity of SFP+, XFP, CFP, and CXP modules can be observed using the configuration interface.

Performance. Although our development board and the AE cores have been designed to support a 100 Gbit/s communication, real-world experiments have so far only been undertaken using a 40 Gbit/s ciphertext interface due to financial reasons[4]. Nevertheless, measurements of the overall system proved it to be operational at data rates up to 40 Gbit/s with all the features described above. The whole digital system showed a constant latency of 3.5 µs for all 10 Gbit/s Ethernet links when configured with OCB-Serpent. Correct transmission of frames up to a length of 16,000 Bytes was observed. The total power dissipation of the overall development board is 45 W, thereof 14 W are consumed solely by the FPGA. If configured with OCB-Serpent, the FPGA's utilization corresponds to 32 %. Note that when operating in the 40 Gbit/s configuration, only two fully unrolled encryption cores for the transmitting path and two fully unrolled decryption cores for the receiving path are required.

[4] The 100 Gbit/s CFP module is about eight to ten times more expensive than the 40 Gbit/s module.

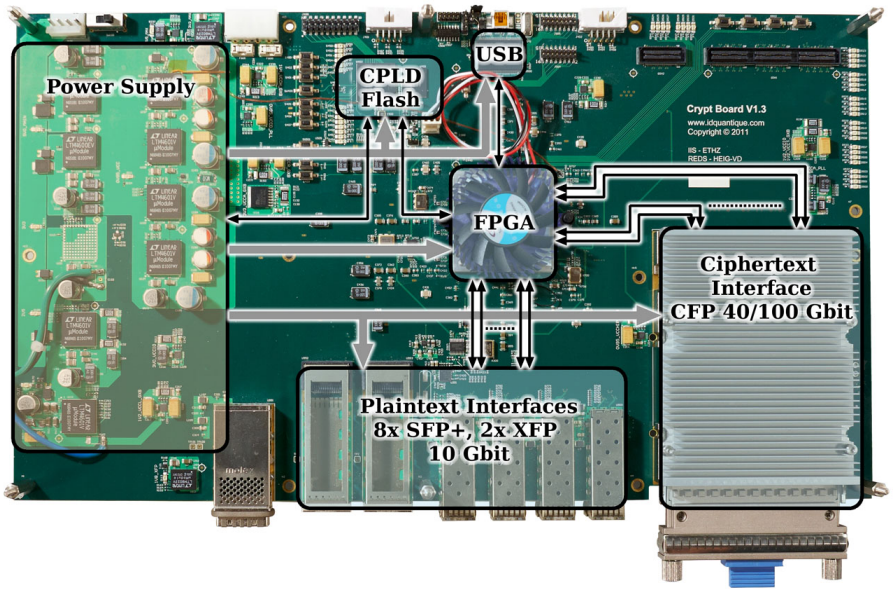

Fig. 6. Overview of the 100 Gbit/s AE development board

6.2 PCB Design

Designing such a complex system in one shot is, in our opinion, not a sound engineering strategy. While reconfiguration of the FPGA does not pose a problem at all, the design of the underlying PCB is "rather statical". Therefore, we have adopted a two-stage design process where we have developed two PCBs. The first system shown in Fig. 6, with its main features listed in Table 3, was used as a development board with all the main components and allowed us to test the basic functionality. The second PCB is the final prototype, and in addition to fixing problems detected in the first design, also adds a number of changes to meet the industrial requirements.

In this section, we will describe the main problems (power distribution and signal quality) we have encountered while designing the development system and will also explain the optimizations we have performed for a follow-up board.

Power Distribution. The first challenge in the system design is establishing the connections to the FPGA which uses a 1932-pin BGA package. While the signal connections offer a formidable challenge in terms of routing, the real problem is in power routing. The system uses in total 14 different power supplies and the main digital power supply of 0.95 V was estimated to consume as much as 48 A. The only practical solution to supply the FPGA with stable power is using several dedicated low-impedance power planes. As a result of these considerations, a 24-layer stack was designed for the development PCB. Even though we

Table 3. Main components of the development board for the 100 Gbit/s AE system

Component	**Type**	**Description**
Networking and encryption engine	Altera Stratix IV GT	FPGA model EP4S100G5F45 with high-speed transceivers, 1932-BGA
Board controller	Altera MAX II	CPLD EPM2210F256
Plaintext interfaces	8 SFP+	10 Gbit/s Ethernet interfaces. Short range (4 SFP+ prepared for Fibre Channel)
	2 XFP	10 Gbit/s Ethernet interfaces. Medium rage
Ciphertext interfaces	1 CFP	40 Gbit/s or 100 Gbit/s wavelength multiplexing four or ten 10 Gbit/s electrical streams per direction
	1 CXP	100 Gbit/s active cable using ten 10 Gbit/s fibres per direction. Short range
USB interface	Cypress EZ-USB FX2LP	For configuration, key transfer, and statistics
Power system	4 LMT4601	4-phase 0.95 V FPGA core supply. max. 48 A
	6 further switched regulators	Digital supplies and analog pre-supplies
	13 linear regulators	Analog and timing block supplies
Clocking system	7 oscillators	System clocks, Transceiver clocks, 24 ... 644 MHz
PCB	NELCO NP400-13EP	24 layers, 387 mm × 220 mm × 3 mm, 1175 components

had anticipated problems with the 0.95 V supply, measurements showed that the voltage-drop across the power plane was still too high. Therefore, we have added two additional power planes for the final design.

Power considerations have also dictated the organization of the layer stack. In the development board, high currents were concentrated on thicker low-impedance layers in the uppermost layers, close to the energy hungry components and their blocking capacitors. However, this asymmetric PCB stack could not be used for the second PCB, since a different manufacturer had to be used. As a result, half of the power layers had to be moved to the bottom for the second PCB.

Signal Quality. As expected, routing several 10 Gbit/s high-speed differential lines across a large PCB turned out to be a challenging task. In total, the development board used sixty impedance-matched differential lines and ten differential clock signals in the frequency range of 156 to 644 MHz. In the development board, these signals were routed on dedicated high-speed layers towards the bottom of the layer stack, where signal quality was not further compromised by longer via stubs. Although utmost care was taken in the design of these differential lines, actual measurements on the board revealed that the attenuation on high-speed signals was critical and problems were detected in impedance matching of the vias. We were able to reach the bit error rates specified in the IEEE 802.3ba standard by programming the transceivers on the FPGA side.

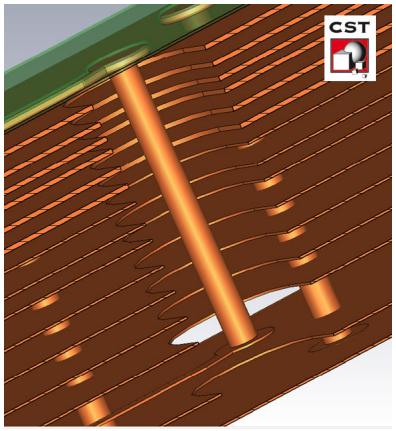

Fig. 7. 3D simulation of a differential via from an impedance-matched wire on top to a wire of layer 14 of the pad stack

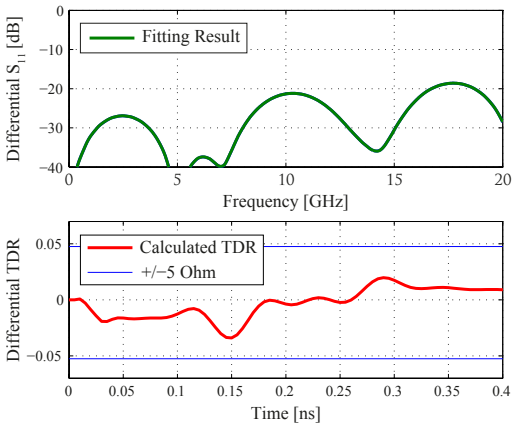

Fig. 8. Reflexion coefficient S_{11} of the simulated differential via in Fig. 7 and simulated, differential time-domain reflectometry (TDR) result

As mentioned earlier, for the second PCB a new manufacturer had to be used, which necessitated a change in the layer stack. In the new layer stack, the high-speed signals are placed in the center of the stack, sandwiched between ground planes. We created a 3D model of the new layer stack using CST Microwave Studio as seen in Fig. 7. This allowed us to make detailed simulations on the behavior of differential vias and determined the best possible geometry to be used. Fig. 8 gives the result of the S_{11} parameter and simulated TDR behavior of the differential via shown in Fig. 7.

Advanced PCB. By applying a two-stage design approach, we achieved the following goals. First, the design constraints for the initial board are relaxed, allowing the board to be manufactured early in the process. Second, the development board is then actively used throughout the development of the AE core and the surrounding system, allowing real measurements on a representative system. These in turn were used to identify problems in the development board and has guided the design of the final PCB. In addition to the weaknesses detected in the first PCB design, it was decided to make the following changes to meet industrial constraints:

- Added two additional layers to improve power distribution.
- Moved the secret key port from USB to PCIe to improve throughput.
- Removed the CXP active cable interface which was deemed to be unnecessary for the application.
- Replaced the two XFP modules by two SFP+ modules.
- Added electronic dispersion compensation (EDC) to six of the SFP+ modules.

– Adapted the dimensions of the PCB to better comply with the requirements of the industrial partner.

However, our two-stage approach also had some drawbacks. The design parameters for such complex PCB systems are not standardized, and most of these parameters need to be negotiated with the PCB manufacturer directly. If for some reason, the PCB manufacturer has to be changed, it is likely that the chosen parameters can not be reused, necessitating time-consuming re-design work. In our case, we were forced to change manufacturers as the initial manufacturer filed for bankruptcy. It proved to be quite difficult to find an alternative PCB manufacturer, 12 out of the 14 companies we have contacted refused the design due to its high complexity.

The advanced PCB now has 26 layers and is 3.3 mm thick. This thickness posed additional challenges for drill holes. The maximum practical aspect ratio (thickness/diameter) for drill holes is around 16:1. Therefore, the thickness of the board directly determines the minimum diameter of the vias that can be used. Since smaller vias are required for impedance matching of 10 Gbit/s differential lines, a delicate balancing act is necessary to reconcile the demands of signal quality on one hand and safe manufacturability on the other hand.

As a consequence of both the increased number of layers, and the change of the manufacturer, we were no longer able to use vias that were 0.22 mm in diameter, but had to adjust the minimal via size to 0.25 mm. This change alone required significant re-design effort.

7 Conclusion

In this work, we described a hardware architecture for high-speed authenticated encryption (AE) using block ciphers on FPGAs, based on *alternative* cryptographic primitives. Our design operates in the offset codebook (OCB3) mode of operation and contains four parallel Serpent block cipher cores for the encryption part in order to achieve the desired data rates according to IEEE 802.3ba. The OCB3-Serpent architecture reaches a throughput of 141 Gbit/s and thus, outperforms all GCM-AES implementations on FPGAs available to date. Although OCB3 has not (yet) been approved by the NIST, its structure (as well as that of Serpent) is definitely suitable for high-throughput implementations as shown during this work. Moreover, we present a custom-designed printed circuit board for the Stratix IV FPGA, which allows us to use the presented AE schemes in real-world applications processing data with up to 100 Gbit/s.

Acknowledgement

This work is part of the QCRYPT project, which is evaluated by the Swiss National Science Foundation and financed by the Swiss Confederation with funding via Nano-Tera.ch.

The authors would like to thank the entire team from the Microelectronics Design Center at the ETH Zurich for their help during the development of the printed circuit board as well as Christian Pendl from Graz University of Technology for his contributions to the digital part of the system.

References

1. IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment 4: Media Access Control Parameters, Physical Layers and Management Parameters for 40 Gb/s and 100 Gb/s Operation. IEEE Std 802.3ba-2010 (Amendment to IEEE Standard 802.3-2008) pp. 1–457 (22 2010)
2. Anderson, R., Biham, E., Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. In: Proceedings of the First AES Candidate Conference. National Institute of Standard and Technology, Ventura, CA, USA (Jun 1998)
3. Biham, E., Anderson, R., Knudsen, L.: Serpent: A New Block Cipher Proposal. In: Vaudenay, S. (ed.) Fast Software Encryption, Lecture Notes in Computer Science, vol. 1372, pp. 222–238. Springer Berlin / Heidelberg (1998)
4. Crenne, J., Cotret, P., Gogniat, G., Tessier, R., Diguet, J.P.: Efficient key-dependent message authentication in reconfigurable hardware. In: Field-Programmable Technology (FPT), 2011 International Conference on. pp. 1–6 (Dec 2011)
5. Dworkin, M.: Recommendations for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. Tech. rep., NIST (2004)
6. Dworkin, M.: Recommendations for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Tech. rep., NIST (2007)
7. Henzen, L., Fichtner, W.: FPGA Parallel-Pipelined AES-GCM Core for 100G Ethernet Applications. In: ESSCIRC, 2010 Proceedings of the. pp. 202 –205 (Sep 2010)
8. IEEE Std 802.11-2007: IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (Jun 2007)
9. Jutla, C.: Encryption Modes with Almost Free Message Integrity. In: Pfitzmann, B. (ed.) Advances in Cryptology — EUROCRYPT 2001, Lecture Notes in Computer Science, vol. 2045, pp. 529–544. Springer Berlin / Heidelberg (2001)
10. Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: Joux, A. (ed.) Fast Software Encryption, Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer Berlin / Heidelberg (2011)
11. McGrew, D.A., Viega, J.: The Galois/Counter Mode of Operation (GCM). NIST Modes Operation Symmetric Key Block Ciphers (2005)
12. Muehlberghuber, M., Keller, C., Felber, N., Pendl, C.: 100 Gbit/s Authenticated Encryption Based on Quantum Key Distribution. In: 2012 IEEE/IFIP 20th International Conference on VLSI and System-on-Chip (VLSI-SoC). pp. 123–128 (Oct 2012)
13. Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., Roback, E.: Report on the Development of the Advanced Encryption Standard (AES). Tech. rep., National Institute of Standards and Technology (NIST) (2000)

14. NIST: Advanced Encryption Standard (AES) (FIPS PUB 197). National Institute of Standards and Technology (Nov 2001)
15. Rogaway, P.: OCB Free Licenses (2013), `http://www.cs.ucdavis.edu/~rogaway/ocb/license.htm`, [Online; accessed 05-March-2013]
16. Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In: ACM Conference on Computer and Communications Security. pp. 196–205 (2001)
17. Viega, J., Mcgrew, D.: The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106 (Proposed Standard) (Jun 2005)
18. Wang, J., Shou, G., Hu, Y., Guo, Z.: High-speed architectures for GHASH based on efficient bit-parallel multipliers. In: Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on. pp. 582 –586 (2010)
19. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). RFC 3610 (Informational) (Sep 2003)
20. Zhou, G., Michalik, H., Hinsenkamp, L.: Improving Throughput of AES-GCM with Pipelined Karatsuba Multipliers on FPGAs. In: Becker, J., Woods, R., Athanas, P., Morgan, F. (eds.) Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, vol. 5453, pp. 193–203. Springer Berlin / Heidelberg (2009)

A OCB Algorithms

Algorithm 2 lists the calculation of the table values $L[..]$ required during the OCB mode of operation. The *double*-procedure is defined according to:

$$double(X) = (X \ll 1) \oplus (msb(X) \cdot 0x87),$$

where $msb(X)$ represents the most significant bit of X using binary representation.

Algorithm 2 Table value calculations.

Input: Cipher key K, Number of message blocks m
Output: $Setup(K, m)$

1: $L_* \leftarrow E_K(0^{128})$
2: $L_\$ \leftarrow double(L_*)$
3: $L[0] \leftarrow double(L_\$)$
4: **for** $i = 1$ to $\lfloor log_2(m) \rfloor$ **do**
5: $\quad L[i] \leftarrow double(L[i-1])$
6: **end for**
7: **return** $L_*, L_\$, L[0] \ldots L[\lfloor log_2(m) \rfloor]$

The initial offset Δ is determined according to Algorithm 3. $X \ll i$ denotes a left shift operation of X by i bits.

Algorithm 4 describes the calculation of $Hash_K(A)$. Since the *Setup* procedure already gets called during the actual encryption process of OCB (cf.

Algorithm 3 Initial offset (Δ) calculation.

Input: Nonce N, Message block length n, Cipher key K
Output: $Init(N, n, K)$

1: $Bottom \leftarrow N \wedge 1^6$ ▷ $Bottom$ = Least six LSBs of N
2: $Nonce \leftarrow 0^{127-|N|}||1||N$
3: $Top \leftarrow (1^{122}||0^6) \wedge Nonce$ ▷ Zeroing out least six LSBs of $Nonce$
4: $Ktop \leftarrow E_K(Top)$
5: $Stretch \leftarrow Ktop||(Ktop \oplus (Ktop \ll 8))$
6: $\Delta \leftarrow (Stretch \ll Bottom) \wedge 1^n$ ▷ Use first n bits of $Stretch \ll Bottom$
7: **return** Δ

Algorithm 1, line 4), line 3 in Algorithm 4 can be omitted as long as the table values $L[..]$ are globally available.

Algorithm 4 Authentication hash ($Hash_K(A)$) calculation.

Input: Associated data A, Associated data block length q, Cipher key K
Output: $Hash_K(A)$

1: $\{A_1, \ldots, A_p, A_*\} \leftarrow A$, with $|A_i| = q$ and $|A_*| < q$
2: $Sum \leftarrow 0^{128}$; $\Delta \leftarrow 0^{128}$
3: $L_*, L[0] \ldots L[\lfloor log_2(p) \rfloor] \leftarrow Setup(K, p)$
4: **for** $i = 1$ to p **do**
5: $\quad \Delta \leftarrow \Delta \oplus L[ntz(i)]$ ▷ $Inc_i(\Delta)$
6: $\quad Sum \leftarrow Sum \oplus E_K(A_i \oplus \Delta)$
7: **end for**
8: **if** $A_* \neq \varnothing$ **then**
9: $\quad \Delta \leftarrow \Delta \oplus L_*$ ▷ $Inc_*(\Delta)$
10: $\quad Sum \leftarrow Sum \oplus E_K(A_*10^* \oplus \Delta)$, with
$\quad\quad A_*10^* = A_* \| 1 \| 0 \ldots 0$, such that $|A_*10^*| = q$
11: **end if**
12: **return** Sum