A High-Throughput Hardware Implementation of NAT Traversal for IPSEC VPN

Tran Sy Nam[1], Hoang Van Thuc[1] and Nguyen Van Long[1]

[1] Institute of Cryptographic Science and Technology, Vietnam Government Information Security Committee, Vietnam

***Abstract*:** In this paper, we present a high-throughput FPGA implementation of IPSec core. The core supports both NAT and non-NAT mode and can be used in high-speed security gateway devices. Although IPSec ESP is very computing intensive for its cryptography process, our implementation shows that it can achieve high throughput and low lantency. The system is realized on the Zynq XC7Z045 from Xilinx and was verified and tested in practice. Results show that the design can gives a peak throughput of 5.721 Gbps for the IPSec ESP tunnel mode in NAT mode and 7.753 Gbps in non-NAT mode using one single AES encrypt core. We also compare the performance of the core when running in other mode of encryption.

***Keywords*:** IPSEC, ESP, NAT, Zynq, FPGA, IKE.

1. Introduction

Networking security has become a crucial issue in the Internet today, since the fast-growing network has suffered enormous amount of attacks everyday. Attacks on the commercial Internet can be various from eavesdropping, data modification, spoofing to repudiation of transactions, which can cause serious loss. In order to address this security issue the IPSec protocol [2, 3] was developed.

The IP Security (IPSec) protocol is one of the most popular protocols used today to provide confidentiality, authenticity and integrity to internet communications. It supports many cryptographic algorithms to provide the security needed for the network users. Once established, IPsec allows transparently protect all application traffic and network services that makes use of IP packets.

IPSec protocol consists of two parts, IKE protocol and ESP protocol [2]. Projects such as Kame [6], Openbsd [7], Freeswan [8], OpenIKEv2 [9] and strongswan [10] provide software implementations of IKE protocol, while ESP protocol is often a part of operating systems. Many operating systems, such as Windows and Linux support ESP in kernel. There is also a full software implementation of IPSec in user space like Strongswan [10], Rockhopper [11]. These software implementations of IPSec ESP, both in user space and kernel space, generally can not gain high throughput and low latency in comparision with hardware implementations. IPsec software implementations can also be improved by using hardware accellerators to offload crypto operations or IPsec operations to hardware [12, 13, 14].

The most well-known application of IPSec is Virtual Private Network (VPN), which often uses IPSec protocol in tunnel mode. For many large enterprises and companies, the internet connection is now more critical than ever. Not just emails and web browsing are used for business, IoT solutions [1], cloud computing or cluster computing nowadays impose heavy demands for higher speed. Since there is a tradeoff between security levels and the performance requirements in the networks, network users have to choose relevant cryptographic algorithms to gain required bandwidth to meet these demands. To tackle the higher-throughput and better security problems, specialized hardware VPN appliances are used. Cisco and Juniper are leading companies in this field, their network equipments such as firewall or routers are often integrated with VPN solution. These commercial devices are capable of handle Gigabits traffic over minimum delay. To achieve this, a large degree or entire IPSec processing operation have to be implemented in hardware (e.g., ASICs).

In recent years, Field Programmable Gate Arrays (FPGAs) have become a preferable alternative for pure software and ASIC solutions. The advantages of FPGA such as flexible architecture and high performance [15] of FPGA make it suitable for cryptographic algorithms. Many works have concentrated on improving speed of AES block cipher [16, 17, 18, 19] or SHA hashing algorithm [20, 21, 22] using FPGA. Taking the benefit of FPGA, System on Chip is a powerful technology that bring software and hardware together in one system in order to solve many complex tasks. The use of SoC is also increasingly demanding as embedded devices are smaller, more flexible and required high performance. Zynq is a SoC product from Xilinx that combine ARM processors with FPGA chip, thus enable highly differentiated designs for a wide range of applications including network security.

The use of NAT together with IPSec has become popular in most today IPSec VPN products. However little work has been done on the hardware implementation of NAT traversal for IPSec. The novelty of this work is the high-throughput FPGA implementation of NAT traversal solution for IPSEC VPN. The maximum NAT traversal throughput the core can achieve is 5.721 Gbps for IPSec AES-256-GCM-16 tunnel mode at a clock rate of 185 MHZ.

Our contribution is not just the IPSEC core itself but a complete system design with hardware and software components running on a modern System on chip Zynq XC7Z045 from Xilinx. All softwares run in embedded Linux ported for ARM including complex IKEv2 protocol. The IPSec ESP protocol is done in FPGA with the aim to increase throughput and decrease latency. To achieve this, the data path of packets has to be realized totally in hardware. This is a non-trivial task since we have to implement packet filtering and packet routing in FPGA as well. In order to implement efficiently in hardware, we choose GCM and CTR as modes of operation for AES-256 block cipher. The core can run on two different cipher suite AES-256-GCM-16 and AES-256-CTR-HMAC-SHA256. The system can act as a high speed IPSec security gateway and was verified and tested in practice.

2. Related Work

In [37] (2018), the authors presented a complete IPSec implementation, both IPSec-AH and IPSec-ESP protocols, each with transport and tunnel mode operation. They use AES in CTR mode for IPSec-ESP and SHA-3 for IPSec-AH. The performance of IPSec-ESP tunnel mode was measured without packet integrity. Moreover, the fact that they used manual key configuration instead of IKE protocol can lead to security problem. One important aspect of CTR mode is that IV or nonce should never be reused with the same key. That's why statically configured keys are not recommended for this mode. In our design, we use IKEv2 to establish fresh keys after the lifetime is reached to ensure this never happens. We also implement a full-feature IPSec-ESP tunnel mode with packet integrity and NAT traversal.

In [36] (2013), the authors presented a multi-core architecture to implement IPSec protocol. This multi-core architecture can be configured with the number of AH/ESP cores and AES-HMAC-SHA-1 cores to achieve high throughput. However their results were only the synthesised results of the design on a Virtex-5 FPGA with the estimated low frequency of 100 MHz.

In [23] (2005), the authors proposed an IPSec implementation on Xilinx Virtex-II Pro FPGA. They moved the key management protocol into the software that runs on the PowerPC. The IPSec protocol was implemented using a softcore processor while encryption and authentication algorithms were performed in hardware. The cryptographic algorithms AES-CBC and HMAC-MD5/HMAC-SHA1 were rather obsolete and the performance of their implementation was quite poor. In [24] (2011), the authors propose an architecture for implementing IPSec on a Xilinx Virtex-4. The proposed solution is based on Partial Reconfiguration technique. They use Round Robin scheduling algorithm to switch between Encapsulating Security Payload (ESP), Authentication Header (AH) and Internet Key Exchange (IKE) in hardware. However, this approach comes with a time delay for switching the crypto core, thus does not allow for extremely high throughput in a typical setting.

In [25] (2008), an IPSec implementation on board ML410 is presented. The design uses AES128 in CBC mode as encryption algorithm and AES-XCBC-MAC-96 as integrity algorithm. The IPsec gateway bases on hardware for timecritical operations like data encryption, network filtering, and packet routing. It uses a lot of hardmacros provided by Xilinx Virtex-4 FX-series FPGA like memories, media access controllers, and the embedded PowerPC CPU. The performances they provided were only of crypto cores, not the whole IPSec design.

3. Technical Background

3.1 IPSec overview

IPSec was designed to secure data at Layer 3 (IP Network Layer). RFC 4301 [3] lists the following services that IPSec provides:

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

IPSec has two security protocols, IP Authentication Header (IPSec AH) [4] and IP Encapsulating Security Payload (IPSec ESP) [5]. AH uses mainly for connectionless integrity and data origin authentication, while ESP supports confidentiality (encryption), connectionless integrity, data origin authentication, and anti-replay integrity. ESP is preferred than AH, since it covers all features of AH. In this work, we focus on ESP protocol.

IPSec also defines Security Association (SA) as a oneway relationship between a sender and a receiver who apply IPSec services to their communication. The SAs have information about a given IPSec communication session, such as protocols, algorithms, session keys, sequence numbers and are stored in Security Association Database (SAD). Another important database in IPSec is the Security Policy Database (SPD). Each entry in the SPD is a security policy that decide on traffic protection using IPsec. There are two ways to establish keys for IPSec, that is manual configuration or automatically configuration. Automatic key configuration protocol is called IKE, it authenticates both peers, sets up a secure channel for key exchange, and negotiates SAs.

Figure 1 depicts the ESP format. The ESP header consists of Security Parameter Index and Sequence Number. SPI is used to determine the keys and parameters in SAD. It helps sender and receiver to agree on the same keys. The Sequence Number is a monotonically increasing counter value, which provides an anti-replay function. The Next Header field specifies the type of the transport protocol used in the upper layer. The Pad Length indicates the number of pad bytes immediately preceding this field. This field is useful for mode of block cipher which is required multiple of the block size such as CBC.

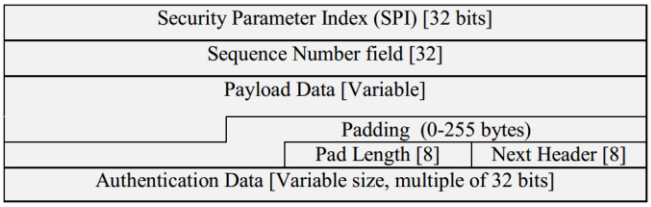

Figure 1. ESP packet format

IPsec can be deployed in either transport or tunnel operation mode. For transport mode, the ESP header is inserted into the IP packet immediately prior to the transport-layer header (e.g., TCP, UDP, ICMP), and an ESP trailer (Padding, Pad Length, and Next Header fields) is placed after the IP packet as Figure 2.

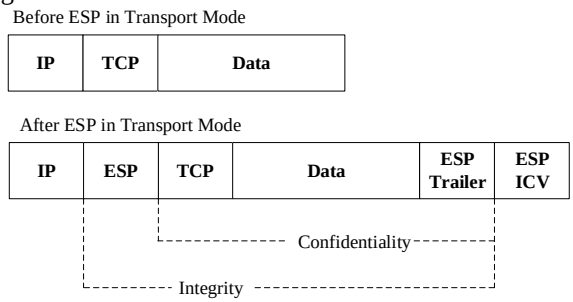

Figure 2. ESP in transport mode

In tunnel mode, entire IP packet is encrypted and authenticated including IP header. Because original IP header is encrypted, intermediate routers would be unable to process such a packet. Therefore, it is necessary to encapsulate new IP header, that will contain sufficient

information for routing but not for traffic analysis, to the encrypted original IP packet. ESP header is inserted into the encrypted IP packet, and an ESP trailer is placed after the encrypted IP packet as Figure 3.

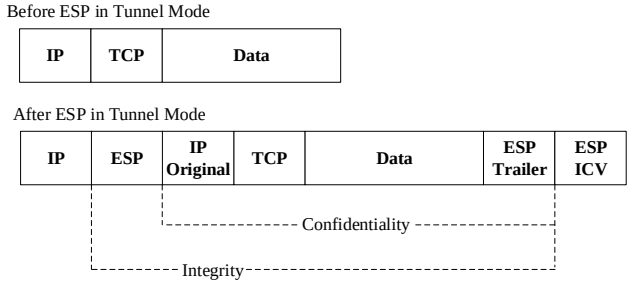

Figure 3. ESP in tunnel mode

The ICV (Integrity Check Value) is computed first at the transmitter by the use of a common authentication algorithm that is also known to the receiver. Then ICV is recomputed at receiver and compared to match the received value for authentication integrity. The ICV computation involves only ESP packet but not IP header, since the IP header may be modified by router (decrease TTL, recompute checksum).
In this paper, we choose ESP tunnel mode since it provides full security services as compared with other modes.

3.2 NAT Traversal

Due to the limitation of public IPv4 address, the Network Address Translation (NAT) is used to allow hosts inside Local Area Networks access the Internet by sharing the public IP address. Unfortunately, IPSec protocol is incompatible with NAT as it modify IP header and breaks the IPsec's security mechanisms. There are some solutions for NAT traversal working with IPSec like UDP Encapsulation [33], Bound End-To-End Tunnel (BEET) Mode [34], UDP Hole Punching for IKE/IPsec [35].
The UDP Encapsulation is used to implemented NAT traversal feature for IPSec ESP core in this work. The process of UDP encapsulation is shown in Figure 4.

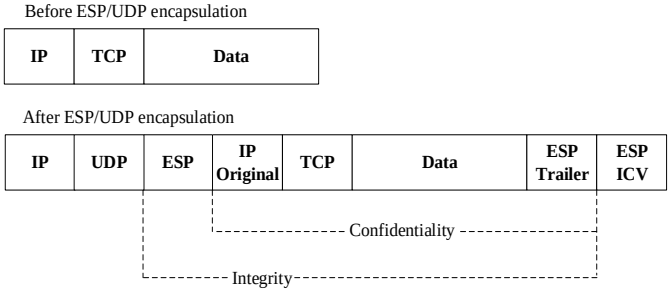

Figure 4. UDP encapsulation

The UDP header is inserted before ESP header in packet after the encryption and authentication. The outer IP header has to update total length, protocol and header checksum fields for new UDP packet. By using UDP packet, the hosts behind a NAT can be detected with port number in UDP header, thus making the ESP tunnel working without affect internal packet headers.

3.3 Zynq architecture

The Zynq®-7000 family is a System on Chip developed by Xilinx. Each Zynq-7000 device has integrates a dualcore or single-core ARM® Cortex™-A9 based processing system (PS) and 28 nm Xilinx programmable logic (PL). The FPGA families such as Virtex-7, Kintex-7, and Artix-7 has to make use of a soft core processor like Xilinx's Microblaze for handling software. By replacing a soft core processor with a hard processor, the flexibility of the processor remains while the performance increases significantly. ARM processor can operate at a higher clock frequency as compared to soft cores, thus enable processing multitasks. Also, the overall cost and physical size of the device can be reduced by integrating FPGA and processor in a single chip.
The FPGAs in PL can be programmed using hardware description language (HDL), making them ideal for performing high-speed logic, arithmetic and data flow subsystems, while the processor in PS supports software routines and/or operating systems. These two components can interconnect with each other using industry standard Advanced eXtensible Interface (AXI) connections.
The Zynq processing system consists of not just the ARM processor, but much more resources such as Application Processing Unit (APU), peripheral interfaces, cache memory, memory interfaces, interconnect, and clock generation circuitry [29]. A block diagram showing the architecture of the PS is shown in Figure 5.

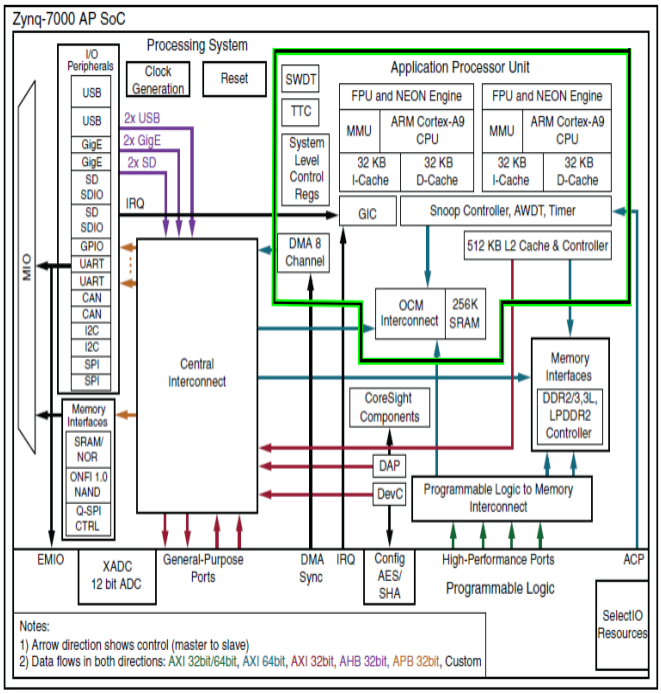

Figure 5. The Zynq Processing System

The programmable logic of Zynq is based on Artix®-7 and Kintex®-7 FPGA fabric. These FPGA often composed of slices and Configurable Logic Blocks (CLBs) and Input/Output Blocks (IOBs) for interfacing, which are the main logic resources for the target circuit.

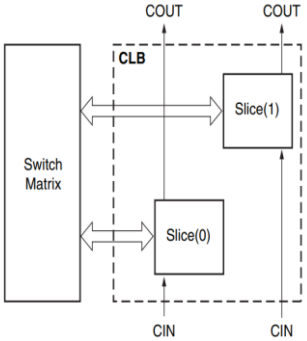

Figure 6. Arrangement of Slices within the CLB

These CLBs are used to implementing sequential as well as

combinatorial circuits. Each CLB element consists of two slices and is connected to a switch matrix for access to the general routing matrix. Four 6-input LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a CLB as Figure 5 [30].

4. Proposed Implementation

The proposed IPSec ESP implementation is shown in Figure 7. The IPSec core composed of 5 components:

- Policy controller
- SA controller
- ESP encapsulation processing
- Cryptographic algorithms
- NAT/non-NAT processing

The processing detail of each component is given separately below.

4.1 Policy controller

The *Policy controller* is implemented as packet filter or access controller list (ACL). It controls the flow of packets based on the fields in packets like IP address, ports. In our design, filtering packets is carried out by checking the matching of source IP address, destination IP address, Protocol in IP header, source UDP/TCP Port, destination UDP/TCP Port.

Policy controller will search in SPD (Security Policy Database) for a policy that match packet. One of three decision will be applied for the packet: BYPASS, DISCARD, or PROTECT. If no policy found in SPD, IPSEC Core will drop the packet and process next packet. If packet match with BYPASS policy, it will be passing out without further processing. Finally, if PROTECT policy is found, the packet will be protected by using IPSec ESP tunnel mode. The default policy of SPD is to drop packets. This policy is considered more secure and easier to maintain than default accept policy. SPD will block any undesired packets, thus prevent denial-of-service attacks.

To search for matching entry in SPD, TCAM (Ternary-Content Addressable Memory) is used. TCAM is a fully associative memory that allows a “don’t care” state to be stored in each memory cell in addition to 0s and 1s [26, 27, 28], that is totally suitable for searching IP address or port.

4.2 SA controller

The main purpose of *SA controller* is to find the keys, parameters from SAD (Security Association Database) for the packet to perform encryption, decryption and packet integrity. It is also used to update the values in SAD when require. For example when increasing sequence number or updating bitmap window for protecting replay attack.

When a packet match policy PROTECT from *Policy controller*, a link to the SAD will be supplied to *SA controller* by SAD_ID. This is an index to SAD (Security Association Database) and is used by SA controller to fetch the keys, parameters for the packets.

For inbound packet, if ESP packet is detected, *SA controller* will search for entry in SAD based on the SPI (Security Parameter Index) in ESP header. Then, packet will go through algorithms for anti-replay and packet integrity checks. The IPCore uses 64 bit Extended Sequence Number (ESN) as default and can be configured to change to 32 bit Sequence Number. When using ESN the receiver has to determine 32 bit higher-order bits of the Sequence Number from 32 bit low-order bits, which is transmitted in ESP header. The process of detecting higher-order bits is carried out together with anti-replay and packet integrity checks according to [5].

4.3 ESP encapsulation processing

Packet after ESP encapsulation process will has the format as Figure 3. The process of ESP encapsulation can be viewed more concretely in Figure 8. It first create new IP header with the source tunnel, destination tunnel IP address and inserted this new header to the packet. Then the ESP header and ESP trailer is created and prepend to the packet. The ESP header size can be different depending on the cryptographic algorithms used in IPSec. In this work, the ESP header, ESP trailer and ICV has the format as Table 1 below.

Table 1. Size of ESP header, ESP trailer and ICV

Fields	Size
Security parameter index (SPI)	32-bit
Sequence number	32-bit
IV	64-bit
Payload	Variable
Padding	0-255 bytes
Padding length	8-bit
Next header	8-bit
ICV	128-bit

4.4 Cryptographic algorithms

The AES block cipher with key length 256 bit is selected for encryption algorithm. Two modes of operation for AES we implement in our core are GCM and CTR. The advantage of these modes is only using encryption algorithm for both encryption and decryption. So the AES hardware resources may be reduced by half since we do not need decryption hardware. The AES core itself is designed based on pipelined architecture as Figure 9.

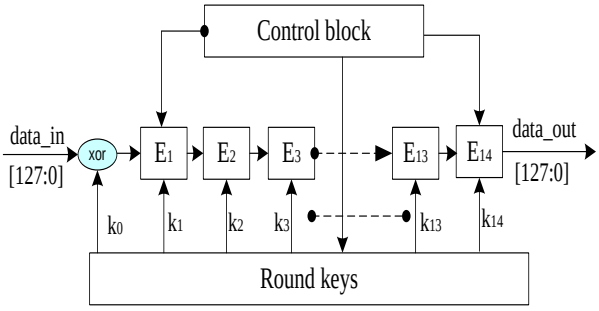

Figure 9. AES pipeline implementation

The pipelined AES generates 128 bit output on each clock cycle. The synthesized results of pipelined AES core using ISE 14.7 is given in Table 2.

To encrypt packet, each 128-bit keystream will be generated by encrypting 128-bit block counter. The format of 128-bit block counter for CTR mode and GCM mode can be found in [31, 32]. Each 128 bit block counter is a combination of 32-bit nonce (CTR) or 32-bit salt (GCM), 64-bit IV, and 32-bit

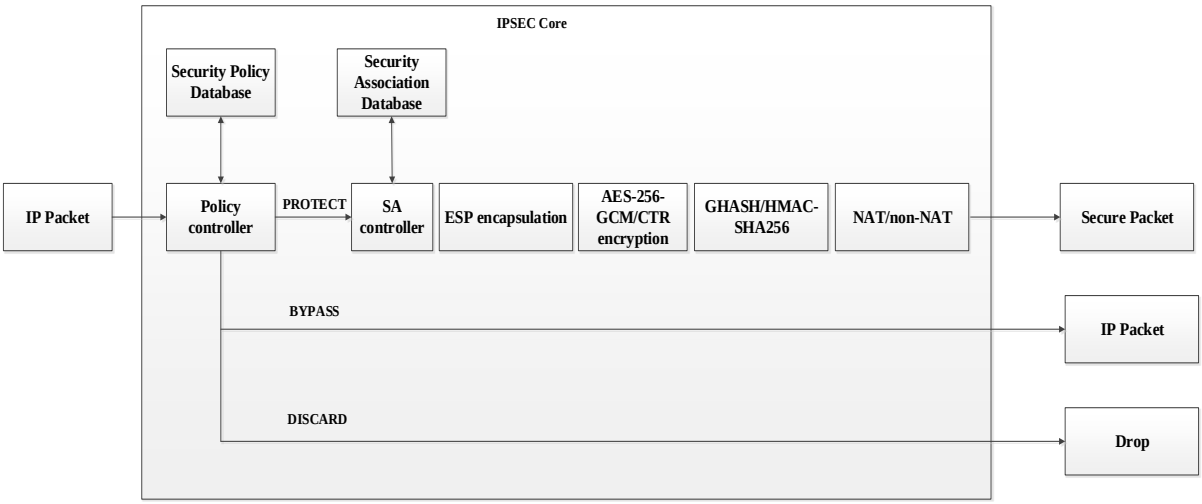

Figure 7. Proposed IPSec Core implementation

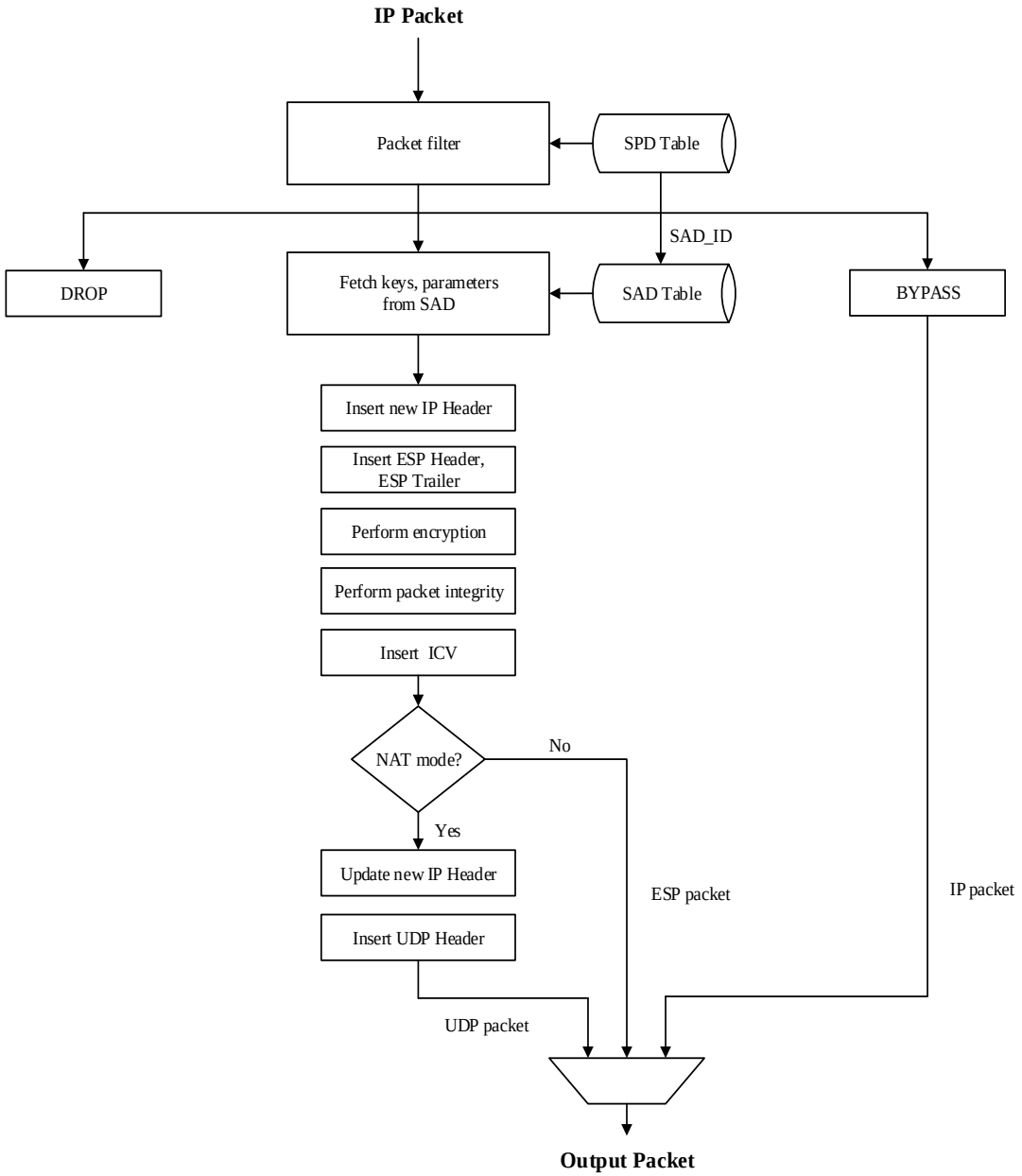

Figure 8. FPGA data flow diagram

counter. The counter is increased by one until the keystream is sufficient to encrypt the whole packet. This process is almost the same for CTR and GCM modes, except that GCM use salt notion instead of nonce.

The nonce/salt is a 32 bit value that is assigned at the beginning of the security association. This value is often generated randomly by IKEv2 when establishing SA. The IV is chosen by the encryptor in a manner that ensures that the same IV value is used only once for a given key. In our implementation, the IV is chosen randomly at the beginning of SA and is XORed with sequence number for each packet. Since sequence number is unique for each packet, so the IV is also unique as well.

We use HMAC-SHA256 as our authentication function for CTR. The SHA-256 core is implemented using the design in [38]. The core can process 512-bit block in 32 clock cycles. The synthesized results of SHA-256 core using ISE 14.7 is shown in Table 2.

GCM use GHASH for authentication mechanism. This algorithm is implemented in our design by using parallel approach of multiplication in $GF(2^{128})$ to keep up with pipelined AES. It take 2 clock cycles to process 128-bit block. The synthesized results is shown in Table 2.

Table 2. Synthesized results of AES-256, SHA-256, GHASH

Algorithm	Max Frequency	Slice	Throughput
AES-256	234 MHz	5526	29 Gbps
SHA-256	154 MHz	1039	2.464 Gbps
GHASH	306 MHz	397	19 Gbps

4.5 NAT/non-NAT processing

If NAT is not presented, the ESP packet is simply send to output without processing. If NAT is detected, the ESP packet will go through UDP Encapsulation as Figure 4. The UDP header is 8 byte length and has the following format.

Source Port	Destination Port
Length	Checksum

Figure 10. UDP header

The source port and destination port in UDP header are given by *SA controller* from SAD. The new IP header has to update total length, protocol and header checksum fields for new UDP packet. After this process, the ESP packet with UDP header can send to endpoint behind NAT without breaking packet integrity.

5. The Complete System Design

As state in [3], a complete IPsec architecture comprises from two part: IKE protocol and ESP protocol. IKE protocol is used to exchange keys securely, while ESP protocol take charge of protecting the main traffic. Our implementation follow the same principal with two components IKE and IPSEC ESP as shown in Figure 11. The IKEv2 protocol automatically negotiate keys for ESP tunnel and must supports two types of Security Associations, IKE SAs and Child Sas. The first type determines how keys are generated, specifying parameters for Diffie-Hellman key exchange, cryptographic, integrity transforms, and pseudo-random functions that generate random numbers for the communication sessions. Child SAs define the sets of encryption and integrity transforms (cipher suite) that are used to protect traffic in IPsec protocols.

Because of the complexity as well as low running frequence, IKE protocol often run in software. Keys created the first time by IKE will be used by ESP protocol utill timeout, then IKE will be invoke again to negotiate new keys.

The IPSEC ESP core is written in Verilog and connect with others core by AXI Stream bus. In order to access the registers in IPSEC Core, AXI interconnect was used to read and write data through memory map. This help IKEv2 software to send keys to IPSEC ESP core after new keys is established.

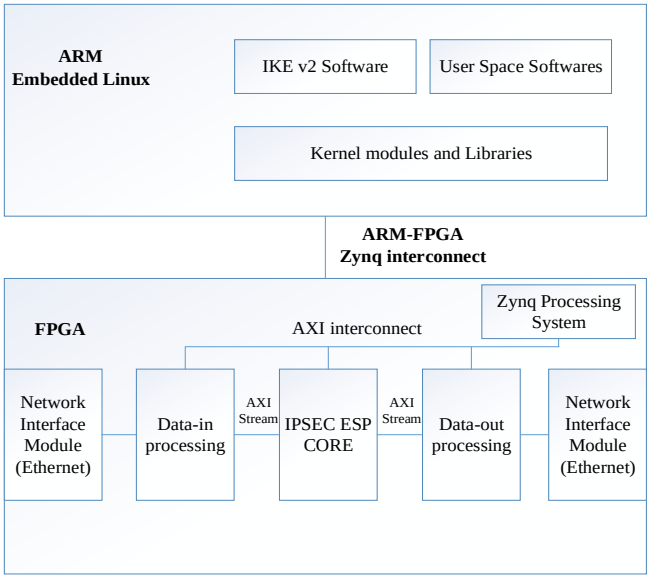

Figure 11. Complete system design

6. Results And Discussion

The IPSec Core was synthesized and implemented using Xilinx Vivado 15.4. The target device is the XC7Z030 and XC7Z045 of Xilinx Zynq-7000 SoC. The implementation results are summarized in Table 3. Moreover, to ensure the compatibility of IPSec core with other implementations, we use Kit ZC706 running embedded Linux with FPGA bitstream loaded from IPSec ESP project. The Kit ZC706 has software and hardware parts as in Figure 11 and can connect to other software implementations of IPSec.

Table 3. Resource summary for IPSec Core

Device utilization	Used	XC7Z030	XC7Z045
LUT	41455	53 %	19 %

The core take 19% of available LUT resources when synthesis for zynq XC7Z045 and 53 % LUT when synthesis for zynq XC7Z030.

For the real maximum operating frequency of the design, we found that the frequency of IPSec Core when using two cipher suites is different.

Table 4. Real maximum operating frequency

IPSec with cipher suite	Maximum Freq
AES-256-CTR-HMAC-SHA256	150 MHz
AES-256-GCM-16	185 MHz

The maximum throughput of the design can be calculated using equation (1):

$$TP = \frac{(Packetsize) * f_{\max}}{Numberof\ clockcycle} \quad (1)$$

Using this equation, we can measure the maximum throughput for IPSec core for different configurations. The packet size of 576 bytes and 1500 bytes are used for the performance results.

Table 5, Table 6 shows the maximum throughput of IPSec core when process packet size of 522 bytes and 1446 bytes in non-NAT mode. The IPSec core adds 54 bytes overhead to the original packet so the ESP packet size is 576 bytes and 1500 bytes.

Table 7, Table 8 shows the maximum throughput of IPSec core when process packet size of 514 bytes and 1438 bytes in NAT mode. The IPSec core adds 62 byte overhead to the original packet so the UDP/ESP packet size is 576 bytes and 1500 bytes.

Table 5. non-NAT mode with 522 byte packet

IPSec with cipher suite	Maximum Frequency	Number of clock cycles	Maximum Throughput
AES-256-CTR-HMAC-SHA256	150 MHz	429	1.460 Gbps
AES-256-GCM-16	185 MHz	160	4.828 Gbps

Table 6. non-NAT mode with 1446 byte packet

IPSec with cipher suite	Maximum Frequency	Number of clock cycles	Maximum Throughput
AES-256-CTR-HMAC-SHA256	150 MHz	926	1.873 Gbps
AES-256-GCM-16	185 MHz	276	**7.753 Gbps**

Table 7. NAT mode with 514 byte packet

IPSec with cipher suite	Maximum Frequency	Number of clock cycles	Maximum Throughput
AES-256-CTR-HMAC-SHA256	150 MHz	467	1.320 Gbps
AES-256-GCM-16	185 MHz	160	4.754 Gbps

Table 8. NAT mode with 1438 byte packet

IPSec with cipher suite	Maximum Frequency	Number of clock cycles	Maximum Throughput
AES-256-CTR-HMAC-SHA256	150 MHz	1022	1.688 Gbps
AES-256-GCM-16	185 MHz	372	**5.721** Gbps

The results show that in NAT mode the maximum throughput of IPSec Core could drop to 26% using AES-256-GCM-16 or 10% using AES-256-CTR-HMAC-SHA256. The UDP encapsulation add 8 bytes UDP header overhead and some processing delay to ESP packets and this leads to decrease performance. This is the first work proposed a complete hardware implementation of NAT traversal for IPSec.

7. Conclusions

In this work, a high throughput IPSec implementation is presented. The IPSec core supports different cryptographic algorithms and is capable of NAT traversal. The system uses IKEv2 running on ARM to negotiate keys for traffic. All ESP process and cryptographic algorithms are performed in hardware, so that they can achieve high performance with little overhead. The results show that our IPSec core can protect the network with high throughput and low latency. The core is compatible with other IPSec system and far more efficient than traditional software implementation.

References

[1]P.R. Kumar, A. T. Wan, and W. S. H. Suhaili “Exploring data security and privacy issues in internet of things based on five-layer architecture,” International Journal of Communication Networks and Information Security, Vol. 12, No. 1, pp. 108-121, 2020.

[2]S. Kent, and R. Atkinson, “Security architecture for the internet protocol,” IETF network working group, RFC2401, 1998.

[3]S. Kent, and K. Seo “Security architecture for the internet protocol,” IETF network working group, RFC4301, 2005.

[4]Stephen Kent, “IP Authentication Header,” RFC 4302, 2005.

[5]Stephen Kent “IP Encapsulating Security Payload,” RFC 4303, 2005.

[6]FreeSwan. [Online]. Available: http://www.freeswan.org.

[7]KAME. [Online]. Available: http://www.kame.net.

[8]OpenBSD. [Online]. Available: http://www.openbsd.org.

[9]Mendez, Alejandro & Fernandez, Pedro & López, Rafael & Martinez Perez, Gregorio & Skarmeta, Antonio & Taniuchi, Kenichi “OpenIKEv2: Design and implementation of an IKEv2 solution,” IEICE Transactions on Information and Systems, Vol. E91.D, Issue 5, pp. 1319-1329, 2008.

[10] Strongswan.[Online].Available: https://www.strongswan.org/.

[11] Rockhopper. [Online]. Available: http://rockhoppervpn.sourceforge.net/.

[12] Intel AES-NI. [Online]. Available: https://www.intel.de/content/www/de/de/architecture-and-technology/advanced-encryption-standard-aes/data-protection-aes-general-technology.html.

[13] Algotronix AES IP-Cores. [Online]. Available: http://www.algotronix-store.com/AES_IP_Cores_s/20.htm.

[14] Chang-Soo Ha, Jong Hyoung Lee, Duck Soo Leem, Myoung-Soo Park, and Byeong-Yoon Choi “ASIC Design of IPsec Hardware Accelerator for Network Security,” In IEEE Asia-Pacific Conference on Advanced System Integrated Circuits (APASIC), pp. 168-171, 2004.

[15] W. Vander, K. Benkrid “High-Performance Computing Using FPGAs,” Springer book, ISBN: 978-1-4614-1790-3.

[16] François-Xavier Standaert, Gael Rouvroy, Jean-Jacques Quisquater, Jean-Didier Legat “A Methodology to Implement Block Ciphers in Reconfigurable Hardware and its Application to Fast and Compact AES RIJNDAEL Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays, pp. 216-224, 2003.

[17] Disha Yadav, Arvind Rajawat “Area and Throughput Analysis of Different AES Architectures for FPGA Implementations,” IEEE International Symposium on Nanoelectronic and Information Systems (iNIS), 2016.

[18] K. Rahimunnisa1 , P. Karthigaikumar1 , Soumiya Rasheed, J. Jayakumar, S. SureshKumar “FPGA implementation of AES algorithm for high throughput using folded parallel architecture,” Security and Communication networks, pp. 2225–2236, 2014.

[19] Soufiane Oukili, Seddik Bri “High throughput FPGA Implementation of Advanced Encryption Standard Algorithm,” TELKOMNIKA, Vol.15, No.1, pp. 494-503, 2017.

[20] C. Xiao-hui and D. Jian-zhi “Design of SHA-1 Algorithm Based on FPGA,” Second International Conference on

Networks Security, Wireless Communications and Trusted Computing, Wuhan, China, pp. 532-534, 2010.
[21] Michail, Harris & Athanasiou, George & Kelefouras, Vasilios & Theodoridis, George & Stouraitis, Thanos & Goutis, Costas "Area-Throughput Trade-Offs for SHA-1 and SHA-256 Hash Functions' Pipelined Designs," Journal of Circuits, Systems and Computers, Vol. 25, No. 4, pp. 1-27, 2016.
[22] H. Michail, G. Athanasiou, A. Kritikakou, C. Goutis, A. Gregoriades and V. Papadopoulou "Ultra high speed SHA-256 hashing cryptographic module for IPSec hardware/software codesign," International Conference on Security and Cryptography (SECRYPT), Athens, Greece, pp. 1-5, 2010.
[23] Jing Lu and John Lockwood "IPSec Implementation on Xilinx Virtex-II Pro FPGA and Its Application," Parallel and Distributed Processing Symposium & 19th IEEE International Proceedings, pp. 158b, 2005.
[24] A. Salman, M. Rogawski, and J.-P. Kaps "Efficient Hardware Accelerator for IPSec Based on Partial Reconfiguration on Xilinx FPGAs," in Proceedings of the 2011 International Conference on Reconfigurable Computing and FPGAs, ser. RECONFIG '11. Washington, DC, USA: IEEE Computer Society, pp. 242-248, 2011.
[25] Wolkerstorfer, Johannes & Szekely, Alexander & Lorünser, Thomas "IPsec Security Gateway for Gigabit Ethernet," Austrochip, 2008.
[26] J. Brelet and L. Gopalakrishnan "Using Virtex-II Block RAM for High Performance Read/Write CAMs," Xilinx XAPP204, 2012.
[27] Zheng, Kai & Hu, Chengchen & Lu, Hongbin & Liu, Bin "A TCAM-based distributed parallel IP lookup scheme and performance analysis," IEEE/ACM Transactions on Networking, pp. 863-875, 2006.
[28] Prithwiraj Das, Ria Pathak, P. Augusta Sophy Beulet "Low Power Implementation Of Ternary Content Addressable Memory (TCAM)," International Journal of Engineering and Advanced Technology, Vol. 9, No. 1, pp. 455-460, 2019.
[29] L. H. Crockett, R. A. Elliot, M. A. Enderwitz and R. W. Stewart "Embedded Processing with the ARM CortexA9 on the Xilinx Zynq-7000 All Programmable SoC," The Zynq Book, 2014.
[30] Xilinx, "7 Series FPGAs Configurable Logic Block", User Guide,[Online].Available: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf.
[31] Housley "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)," RFC 3686, January 2004.
[32] Viega, J. and D. McGrew "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)," RFC 4106, June 2005.
[33] A. Huttunen, B. Swander, V. Volpe, L. DiBurro und M. Stenberg "UDP Encapsulation of IPsec ESP Packets," RFC 3948, 2005.
[34] J. Rosenberg "A Bound End-to-End Tunnel (BEET) mode for ESP," RFC draft 09, 2008.
[35] J. Rosenberg "A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," RFC 5245, 2010.
[36] Niu, Y., Wu, L. and Zhang, X. "An IPSec accelerator design for a 10Gbps in-line security network processor," Journal of Computers, Feb. 2013, Vol. 8, No. 2, pp.319–325, 2013.
[37] Muzaffar Rao, Thomas Newe, Edin Omerdic, Gerard Dooly, Elfed Lewis, Daniel Toal "An efficient implementation of FPGA based high speed IPSec (AH/ESP) core," International Journal Of Internet Protocol Technology, Inderscience Enterprises, Vol. 11, No. 2, pp. 97-109, 2018.
[38] H. Michail, G. Athanasiou, A. Kritikakou, C. Goutis, A. Gregoriades and V. Papadopoulou "Ultra high speed SHA-256 hashing cryptographic module for IPSec hardware/software codesign," International Conference on Security and Cryptography (SECRYPT), pp. 1-5, 2010.