

IPSec Implementation on Xilinx Virtex-II Pro FPGA and Its Application

Jing Lu John Lockwood
Reconfigurable Network Group
Applied Research Laboratory
Washington University
St. Louis, MO 63130
{jl1,lockwood}@arl.wustl.edu

Abstract

In this paper, we propose an IPSec implementation on Xilinx Virtex-II Pro FPGA¹. We move the key management and negotiation into software function calls that run on the PowerPC processor core. On the data path, reconfigurable hardware logic implements time-critical functions for AES encryption and HMAC authentication. In our approach, the fast hardware processing is quasi-independent of the software processing. In traditional hardware systems, it is often the case that fast hardware modules wait for slow softwares to feed input data and retrieve output data. This causes the hardware component to stay in idle and suffer low utilization. Our contribution in this paper is to separate the IPSec data path from the control path, where the hardware has a full control of data processing and invokes the control software only when necessary. We illustrate the use of the IPSec implementation on a reconfigurable network device to secure the control and configuration channel.

1. Introduction

For devices deployed remotely on the Internet, security is critical. Network attacks can be classified as passive attacks or active attacks. The former learns and makes use of the information from the system without the awareness of the system administrator. The latter attempts to change the system's behavior or even paralyze the entire system. Common types of attacks include:

- Eavesdropping: Malicious users can tap the network, copy and analyze the traffic. If they are able to see clear-text control and configuration information, they

may discover the system's weak points through analysis.

- Tamper: Unauthorized users can gain control of the remote device by issuing fake control packets. Such kind of attack may change the normal behavior of the system or paralyze it.
- Replay attack: By passive capture of legitimate traffic and subsequent retransmission, malicious users can change the system behavior by "resetting" the system to some previous state without being noticed.
- Denial of Service (DOS) attack: DOS attack can degrade the system performance by overloading the system with massive traffic. It can cause communication breakdown and put the system out of service.

Security services that provide protection to system resources can be divided into four categories:

- Confidentiality ensures that only the authorized entity can view the content of the protected data. By encrypting with AES or other encryption algorithms, data remains safe.
- Authentication makes sure that the communication entity is the one that it claims to be. It includes peer entity authentication and data origin authentication. Digital signatures generated by public key algorithms such as RSA and El-Gamal are hard to forge.
- Integrity guarantees that data received are exactly the same as the data sent, ensuring that no modification, insertion, deletion or replay occurred during the transmission. Message Authentication Code (MAC) computed by hashing a shared secret along with the message provides integrity on the signed document.
- Access control protects resources from unauthorized use. Security policies at network access points are defined to control who can gain access, and under what conditions access is granted.

¹ This work was supported by a grant from Global Velocity. John Lockwood serves as a professor at Washington University and a consultant to that company.

Secrets, such as encryption keys and hash keys, are essential for the security services defined above. Key exchange protocol, such as IKE, provides a mechanism to negotiate and exchange shared secrets between communication entities.

The general consensus of the importance regarding Internet security led to the development of IPSec. IPSec provides a standard mechanism to secure communications across the Internet. It was designed to be compatible with both IPv4 and IPv6. Many companies, such as Cisco, have implemented IPSec capability in their products. In this paper, we present an implementation of IPSec in transport mode on the latest Xilinx Virtex-II Pro FPGA. In an application example of a reconfigurable network device, we examine the security requirements and challenges faced by the devices when they are remotely deployed on the Internet. We discuss how they benefit from the IPSec-based communication channel.

This paper proceeds as follows: Section 2 describes the IPSec framework and the use of its transport mode in end-to-end secure communication. Section 3 discusses how IPSec can be efficiently implemented in the Xilinx Virtex-II Pro FPGA technology. Section 4 shows the implementation of AES, HMAC-MD5 and HMAC-SHA-1. Section 5 shows how IPSec can provide a secure communication channel for reconfigurable network devices. Finally, we conclude our paper in Section 6.

2. IPSec Framework

IPSec protocols include Authentication Header (AH) and Encapsulating Security Payload (ESP). Both AH and ESP use sequence numbers to protect IP packets against replay attacks. In addition, AH provides data integrity and source authentication, while ESP provides data confidentiality and data integrity. Each protocol can operate in two modes: transport and tunnel modes. The transport mode is used in end-to-end communication where the two end hosts have the IPSec capability. The tunnel mode is typically used when the destination of the packet is different from the gateway router, which provides IPSec services for packets it forwards. In this paper, we focus on IPSec in transport mode for provisioning end-to-end secure communication required by remote network devices. Figure 1 shows the scope of the AH authentication and ESP encryption and authentication for IPv4 packet in transport mode. The AH authentication scope includes the original IP header. This is a feature essential to the applications that demand source authentication. In contrast, ESP is designed to encrypt the upper layer protocol data without source authentication. For applications that need both source authentication and data encryption in addition to data integrity, AH should be used in conjunction with ESP. In this case, ESP authentication can

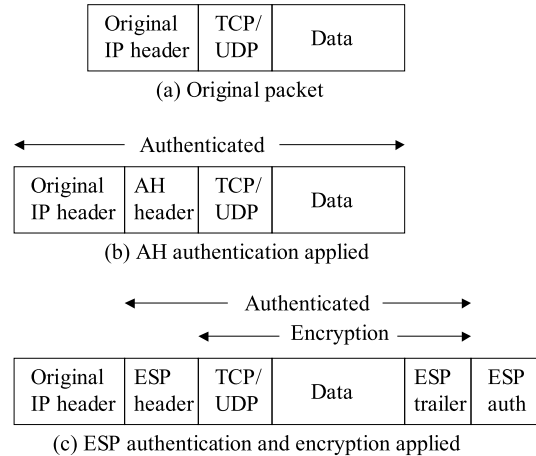


Figure 1: Scope of AH authentication and ESP encryption and authentication for IPv4 packet in transport mode

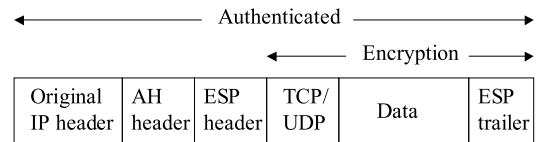


Figure 2: Scope of AH authentication in conjunction with ESP encryption for IPv4 packet in a transport mode

be disabled because AH covers a larger scope of authentication. Figure 2 shows the scope of protection.

IPSec also defines Security Association (SA) as a one-way relationship between a sender and a receiver who apply IPSec services to their communication. SA keeps track of the information about a given IPSec communication session, such as protocols, algorithms, session keys and sequence numbers. An SA is uniquely identified by a randomly chosen number called the Security Parameter Index (SPI) and the destination IP address. Another important concept in IPSec is the Security Policy Database (SPD). Each entry in the SPD defines actions on a subset of IP traffic and has a pointer to an SA for that traffic. Secret key management is done either manually or automatically. Automatic key management protocol is called IKE, which is based on ISAKMP, Oakley and SKEME. Details about SA, SPD and IKE can be found in RFC 2401 - 2409.

In the following subsections, we describe the IPSec processing flows in hardware. AH and ESP are working together to provide source authentication, data encryption and data integrity.

2.1. Outbound Processing

Figure 3 shows the block diagram of the IPSec outbound processing. Traffic comes in from the left. The IP header is passed to the *IPSec controller*, where two table lookups are

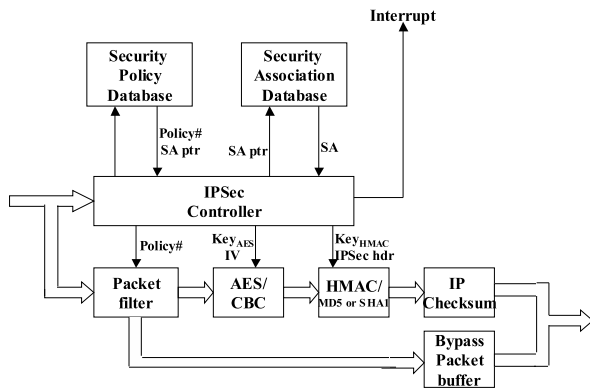


Figure 3: IPsec outbound processing

performed to first retrieve the security policy code and SA pointer, and then use the pointer to obtain the SA associated with the current packet. The policy code passed to the *packet filter* indicates the possible action that the *packet filter* may take: drop the packet, bypass the packet or apply IPsec. A bypassed packet is queued in the *bypass packet buffer* before it is sent out. In the case a packet needs IPsec processing, if the SA entry is valid, the *IPsec controller* passes the AES encryption key and Initialization Vector (IV) for CBC mode to the *AES/CBC* module. In the meantime, the authentication key and AH header are passed to the *HMAC/MD5 or SHA-1* module. Since the process changes the IP packet length, IP checksum needs to be recomputed before the packet can be sent out. A null SA pointer along with the IPsec policy raises an interrupt, indicating the start of a key negotiation process in the software.

2.2. Inbound Processing

The block diagram of the IPsec inbound processing is shown in Figure 4. Unlike the outbound processing, no interrupt will be generated in case of a null SA entry. This is because the SA negotiation is initiated by the traffic source, not the traffic destination. The packet is simply dropped when the associated SA pointer is null.

A received packet comes in from the left. The packet with a wrong IP checksum will be dropped by the *IP checksum verifier*. The *packet filter* retrieves the policy code from the *Security Policy Database*. IPsec packet is passed to the *IPsec controller*, *HMAC/MD5 or SHA-1*, and *AES/CBC* modules simultaneously. The *IPsec controller* looks up the AH SA and ESP SA according to the AH SPI and ESP SPI embedded in the AH and ESP headers. Valid SAs provide keys to the *HMAC* and *AES* modules. Once the decryption is done and hash value is computed, the *integrity check* module compares the computed hash value with the hash value field in the AH header; and a decision is made on whether or not to let the packet pass.

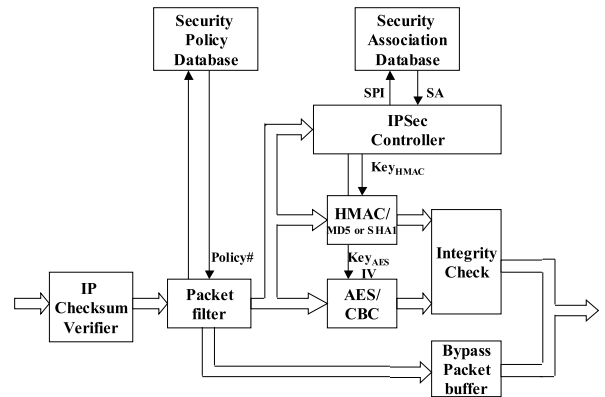


Figure 4: IPsec inbound processing

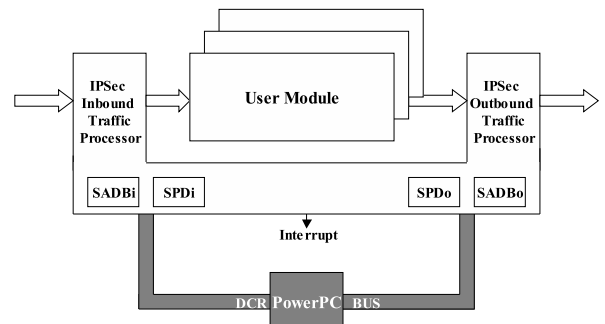


Figure 5: IPsec wrapper to secure user application

3. IPsec on Xilinx Virtex-II Pro

In recent years, Field Programmable Gate Arrays (FPGAs) have achieved sufficient capability to performing complex network processing in programmable hardware. Network devices utilizing FPGAs show a desirable balance between performance and flexibility, which makes FPGA preferable to pure software and ASIC solutions. The Xilinx Virtex-II Pro family incorporates high speed serial transceiver technology and IBM PowerPC 405 hard processor core within a general-purpose FPGA device[6]. Some of the appealing features of Virtex-II Pro Platform FPGAs include:

- RocketIO embedded transceivers with up to 3.125 Gbps baud rate are desirable for high-bandwidth networking and communication applications.
- IBM PowerPC hard processor cores enhance the traditional reconfigurable logic with embedded processing capability.
- Up to 100K logic cells and 8Mb on chip Block RAM provide resources to enable highly complex processing.

The IPsec processing flows described in Section 2 can be implemented on Xilinx Virtex-II Pro FPGA by taking advantage of the embedded processor core, on-chip block

RAM, and reconfigurable logic resources. Figure 5 shows the IPSec protocol wrapper, where the inbound and outbound processing modules are implemented with reconfigurable logic, and SAs and SPDs are implemented using on-chip block RAM. The IPSec outbound processing module generates an interrupt to the PowerPC core every time it detects a packet that needs IPSec processing under the security policy but the corresponding SA is unavailable. Then the IKE daemon running on the IBM PowerPC core begins to negotiate an SA with the packet destination. Once the key exchange process ends and a new SA is successfully created, the new SA is written into the outgoing processing module through the Device Control Register (DCR) bus. DCR bus is also used to update security policy in the IPSec modules. Projects such as KAME[2], OPENBSD[3] and FREESWAN[1] provide open source IKE code. We can port the IKE code to the IBM PowerPC 405 embedded processor.

4. Implementation

IPSec standard allows for the use of multiple encryption algorithms including AES [5]. As illustrated in the IPSec processing flows in Section 2, AES is chosen in this particular implementation because of its efficiency and straightforward hardware implementation.

4.1. AES algorithm and Implementation

The AES algorithm works by using cryptographic keys of 128, 192 or 256 bits to encrypt 128-bit data blocks. Figure 6 describes the AES algorithm. In the figure, k is the number of bits in the key, N_r is the number of iterations (or rounds) performed to complete the encryption or decryption of a single data block. N_r is a function of k and is 10, 12 or 14 for k of 128, 192 or 256 bits, respectively. The *key schedule* expands the original key to N_r+1 roundkeys. The operations performed in each round are shown in the figure. Notice, the last round for encryption does not include the *mix columns* step, while decryption is missing the *inverse mix columns* in its last round.

Figure 7 shows the block diagram of the AES implementation. Unlike other AES implementations in [15], this is a full bloom AES implementation. It not only supports three different key length, it also does AES encryption when *cipher mode* signal is high and AES decryption when *cipher mode* signal is low.

AES uses 16 256-byte substitution boxes (Sboxes) for each encryption round and another 16 256-byte inverse Sboxes for each decryption round. If these Sboxes were implemented using registers, each round could be completed in one clock cycle. However, this would consume a great deal of logic resource. Therefore, we chose to implement

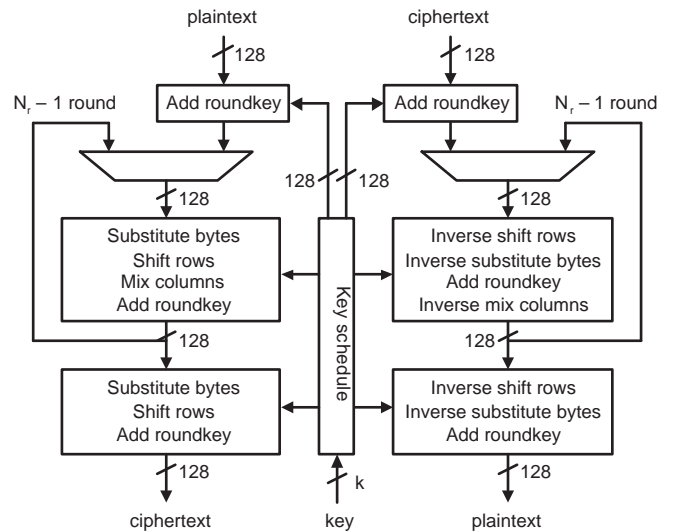


Figure 6: AES Algorithm

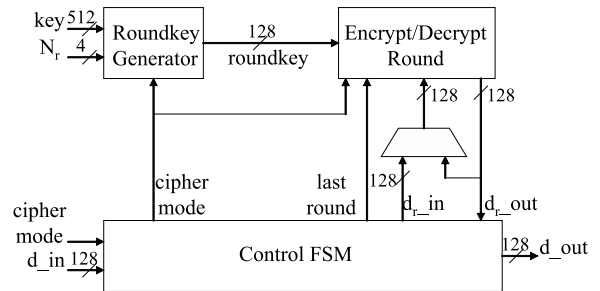


Figure 7: AES Hardware Implementation

these Sboxes in block RAMs. This increased each AES round time to two clock cycles, but we will see later this implementation still yields Gigabit throughput for 128-bit key.

Table 1: AES Synthesis Results on Xilinx Virtex-II Pro 100

	LUTs		BRAM		Timing	Rate
	#	%	#	%	MHz	Mbps
AES-128	2703	3%	44	10%	196.3	1197
AES-192	2710	3%	44	10%	170.9	876
AES-256	2745	3%	44	10%	178.6	778

Table 1 summarizes the synthesis results for AES cores on Xilinx Virtex-II Pro XC2VP100 using Xilinx XST G.31a. In particular, the AES-128 core can process data at 1.2Gbps by using only 3% of the LUTs and 10% of the on chip Block RAM.

4.2. Authentication Algorithms and Implementation

Both MD5 and SHA-1 were implemented as the HMAC hash function core. These algorithms and implementations are described below.

4.2.1. MD5 and SHA-1

MD5 and SHA-1 are message digest algorithms specified for use in IPSec. Both algorithms take as input a message of arbitrary length and produce as output a message digest of 128 bits for MD5 and of 160 bits for SHA-1. The input message is first padded and appended with message length to be multiple of 512 bits. Then the message is processed in 512-bit blocks with an n -bit initial value, where n is 128 for MD5 and 160 for SHA-1. Further details regarding MD5 and SHA-1 can be found at [12] and [4], respectively.

The MD5 and SHA-1 cores were implemented using an iterative architecture and had a latency of 197 and 245 clock cycles respectively for hashing a 512-bit data block. These numbers are high because the four 32-bit additions in each step of the hashing algorithms are spread out over three clock cycles to increase clock speed. The benefit of doing so is to keep the overall system throughput high.

Table 2: MD5 and SHA-1 Synthesis Results on Xilinx Virtex-II Pro 100

	LUTs		BRAM		Timing	Rate
	#	%	#	%	MHz	Mbps
MD5	4050	4%	0	0%	106.6	277.1
SHA-1	5977	6%	0	0%	145.3	303.6

Table 2 shows the synthesis results for the MD5 and SHA-1 cores on Xilinx Virtex-II Pro XC2VP100 using XST G.31a. The throughput of the cores is low because we use an iterative architecture. Since they are sufficiently small, four HMAC-MD5 or HMAC-SHA1 cores can fit within an FPGA and run in parallel to achieve Gigabit throughput.

4.2.2. HMAC

HMAC is used in conjunction with either MD5 or SHA-1. It uses a secret key to validate the information. HMAC can be described by the following equation:

$$HMAC_{text} = H(K \oplus opad, H(K \oplus ipad, text))$$

where K is a secret key, $ipad$ is the byte 0x36 repeated 64 times, $opad$ is the byte 0x5C repeated 64 times, H is the hashing function (MD5 or SHA-1). The results are 128-bit digest for MD5 and 160-bit digest for SHA-1. In the IPSec standard, only the first 96 bits of the digest are used. Figure 8 shows the hardware implementation of HMAC.

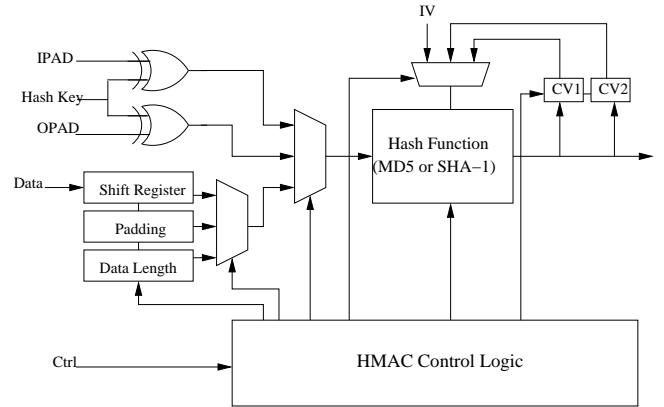


Figure 8: HMAC Hardware Implementation

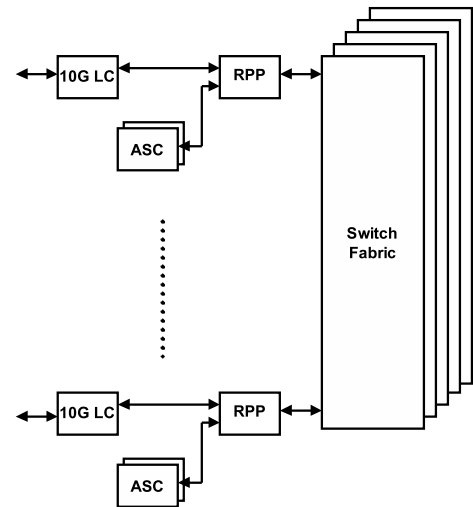


Figure 9: System architecture of the extensible network services platform

The estimated resource usage for one AES-128 core and four HMAC-MD5 cores is 19% in logic cells and 10% in on-chip block RAM for input IPSec processing or output IPSec processing.

5. Application

5.1. The Extensible Network Services Platform

The purpose of the Technologies for Extensible Network Project is to build an extensible network services platform that combines routing and processing capabilities with a single, flexible architecture. We are designing and building this open platform, where fast prototype is enabled to demonstrate and evaluate key components for the next generation of extensible data networks. The overall system architecture is shown in Figure 9.

All ports are linked by a high performance switch fabric. Each port has two major components: Routine Packet

Processor (RPP) and Advanced Service Card (ASC). The RPP performs route lookup, packet classification and queue management. The ASC provides hardware resources needed for advanced network services. Based on the packet classification results, RPP can forward packets to ASC for advanced processing, such as encryption, compression, and content scanning for viruses and worms, etc. Hardware plug-ins can be dynamically loaded into the Virtex-II Pro FPGAs on the ASC to extend the routine capability of network routers[14].

Although the ASC is part of the extensible network router, it is also designed to be used as a standalone programmable network device. An ASC connects to the Internet through two 10 Gigabit Ethernet Line Cards. A central Control and Routing Manager (CRM) is implemented on a Virtex-II Pro FPGA sitting on the ASC. It includes an efficient switch fabric to route traffic among the components on the ASC. Besides, the CRM has a 10/100 Ethernet interface, through which a PC can be connected for local control and monitoring purpose. As an application example, we focus on using ASCs as standalone programmable network devices. Interesting applications developed on the Field-programmable Port Extender (FPX)[9] can be ported to the ASC. These applications include the layered protocol wrappers[7], the extensible system-on-programmable-chip content-aware Internet firewall[10], the content-scanning module for Internet firewall[11], the TCP processor[13] and the deep content inspection using parallel Bloom Filters[8].

Figure 10 shows how an ASC is deployed remotely as a standalone device on the Internet. Control and configuration packets traverse the unreliable Internet before getting to the central control module CRM on the ASC. In particular, hardware plug-ins are loaded dynamically by requirements of user applications. Each plug-in module may have its own need for exchanging control information with other devices or hosts across the Internet. However, plug-in modules either don't have the resources or can't afford to reinforce strong security mechanism on their control traffic. Therefore, central security services such as IPSec implemented on the CRM is desirable and sufficient for the needs of all plug-in modules. IPSec key exchange can be done manually through a local PC connected to the ASC, or automatically through the IKE daemon running on the CRM PowerPC core. Our design will port the open source IKE package to the IBM PowerPC processor core. By strictly following the IPSec standard in our implementation, an ASC is able to communicate with other IPSec capable hosts on the Internet.

Figure 11 shows the architectural overview of the CRM design. The *switch fabric* routes packets among the PowerPC core, two 10 Gigabit line cards and other components on the ASC. For traffic from the Line Cards and destined for the ASC, the *switch fabric* will route it to the

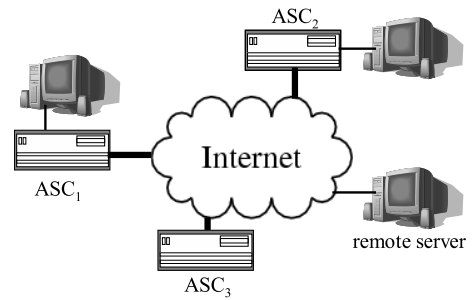


Figure 10: Multiple ASC cards deployed over the Internet

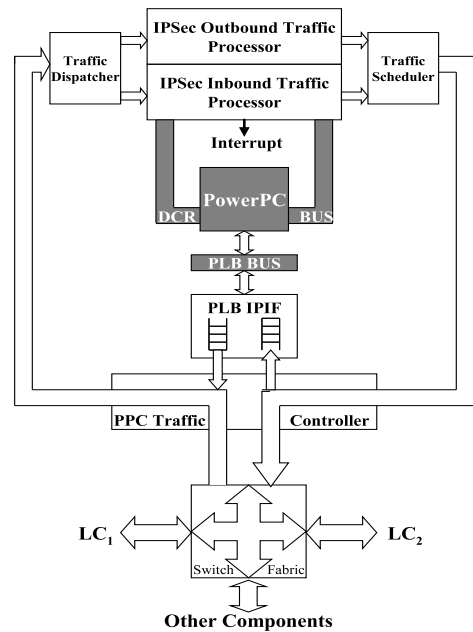


Figure 11: Architectural overview of the CRM design on Virtex-II Pro FPGA

PowerPC side for IPSec inbound processing. According to the security policy, packets may be by-passed, dropped, or processed as IPSec packets. For outgoing traffic originating from other components on the ASC, the *switch fabric* also routes it to the PowerPC side for IPSec outbound processing. Again, packets may be dropped, by-passed, or encrypted and authenticated if IPSec is enabled in the security policy. The IPSec modules also perform IPSec processing on behalf of the softwares running on the PowerPC. In the figure, a customized Xilinx *PLB IPIF* core with two packet FIFOs serves as the interface between the IPSec data path and the PowerPC processor local bus. The *PPC traffic controller* loads control packets out of the PowerPC onto the IPSec data path and unloads packets targeting the PowerPC from the IPSec data path.

6. Conclusion

Secure control and configuration of remote network devices are needed to protect reconfigurable systems from attack. Modern network devices use FPGA technology to provide high performance and programmability. PowerPC processor cores on the latest Xilinx Virtex-II Pro FPGAs enhance the reconfigurable logic with embedded processing capability. By implementing noncritical and resource consuming functions in software running on the PowerPC and time-critical functions in reconfigurable hardware, high overall system performance can be achieved on a signal chip with little overhead.

In this paper, we present an IPSec implementation on Xilinx Virtex-II Pro FPGA. We move the key management and negotiation functions into the software that runs on the PowerPC. The hardware implementation performs the IPSec encryption and authentication functions on the data path with minimum software interference. Our contribution is to separate the data path from the control path, where hardware has a full control of the data processing and invokes the control software only when necessary. Through the implementation of the AES and HMAC-MD5/HMAC-SHA-1 cores, we show that the IPSec modules can fit within a single FPGA and still achieve over 1Gbps throughput. We apply this IPSec implementation to a reconfigurable network device to provide a secure communication channel for remote control and configuration.

7. Acknowledgments

We would like to give special thanks to Dr. Jonathan Turner, project leader of the Technology for Extensible Networks. The design and implementation of the extensible router involves the hard work of other project members: John DeHart, Fred Kuhns and David Zar. We also want to acknowledge Haoyu Song for his earlier contributions to the HMAC implementation.

References

- [1] FREESWAN. Online:<http://www.freeswan.org>.
- [2] KAME. Online:<http://www.kame.net>.
- [3] OPENBSD. Online:<http://www.openbsd.org>.
- [4] FIPSP-180-1, Apr. 1995.
- [5] FIPS-197: Advanced Encryption Standard (AES) . Online:<http://csrc.nist.gov/encryption/aes/>, Feb. 2001.
- [6] Virtex-II Pro Platform FPGA Handbook, Oct. 2002.
- [7] F. Braun, J. W. Lockwood, and M. Waldvogel. Layered Protocol Wrappers for Internet Packet Processing in Reconfigurable Hardware. Technical Report WU-CS-01-10, Washington University in Saint Louis, Department of Computer Science, June 2001.

- [8] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood. Deep Packet Inspection using Parallel Bloom Filters. In *IEEE Micro*, Vol. 24, No. 1, pp. 62-69, Jan. 2003.
- [9] J. W. Lockwood, J. S. Turner, and D. E. Taylor. Field Programmable Port Extender (FPX) for Distributed Routing and Queuing. In *ACM International Symposium on Field Programmable Gate Arrays (FPGA'2000)*, Monterey, CA, USA, Feb. 2000.
- [10] J. W. Lockwood, C. Zuver, C. Neely, J. Moscola, S. Dharmapurikar, and D. Lim. An Extensible, System-On-Programmable-Chip, Content-Aware Internet Firewall. In *Field Programmable Logic and Applications (FPL'2003)*, Lisbon, Portugal, 2003.
- [11] J. Moscola, J. Lockwood, R. P. Loui, and M. Pachos. Implementation of a Content-Scanning Module for an Internet Firewall. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'2003)*, Napa, CA, USA, Apr. 2003.
- [12] R. Rivest. The MD5 Message-Digest Algorithm, RFC 1321. MIT LCS and RSA Data Security, Inc., Apr. 1992.
- [13] D. Schuehler and J. Lockwood. A Modular System for FPGA-based TCP Flow Processing in High-Speed Networks. In *14th International Conference on Field Programmable Logic and Applications (FPL'2004)*, Springer LNCS 3203, Antwerp, Belgium, Aug. 2004.
- [14] D. E. Taylor, J. S. Turner, J. W. Lockwood, T. S. Sproull, and D. B. Parlour. Dynamic Hardware Plugins (DHP): Exploiting Reconfigurable Hardware for High-Performance Programmable Routers. In *Computer Networks*, Vol. 28, Issue 3, pp. 295-310, Feb. 2002.
- [15] I. Verbauwheide, P. Schaumont, and H. Kuo. Design and Performance Testing of a 2.29 Gbps Rijndael Processor. In *IEEE J. SolidState Circuits*, Vol. 38, no. 3, pp. 569-572, 2003.