WLAN Security Processor

Neil Smyth, *Member, IEEE*, Máire McLoone, *Member, IEEE*, and John V. McCanny, *Fellow, IEEE*

Abstract—**A novel wireless local area network (WLAN) security processor is described in this paper. It is designed to offload security encapsulation processing from the host microprocessor in an IEEE 802.11i compliant medium access control layer to a programmable hardware accelerator. The unique design, which comprises dedicated cryptographic instructions and hardware coprocessors, is capable of performing wired equivalent privacy, temporal key integrity protocol, counter mode with cipher block chaining message authentication code protocol, and wireless robust authentication protocol. Existing solutions to wireless security have been implemented on hardware devices and target specific WLAN protocols whereas the programmable security processor proposed in this paper provides support for all WLAN protocols and thus, can offer backwards compatibility as well as future upgrade ability as standards evolve. It provides this additional functionality while still achieving equivalent throughput rates to existing architectures.**

Index Terms—**Cryptography, data security, wireless local area network (WLAN).**

I. Introduction

Wireless devices have limited processing power and battery life. This is contradictory to the ever-increasing data throughputs of complex security protocols, which are in demand in order to continue the growth of wireless technologies [1], [2]. The nature of frequently changing and evolving security protocols also necessitates the use of devices with re-programmable hardware. Such devices support variable functionality to counteract security weaknesses and provide a degree of future proofing for what will inevitably become legacy hardware.

IEEE 802.11i [3] is an optional amendment to the IEEE 802.11 standard offering enhanced security at the medium access control (MAC) layer, through: a) an improved Rivest Cipher 4 (RC4) [4], [5] based scheme for legacy systems; b) advanced encryption standard (AES)-based encryption [6], [7] in newer wireless local area network (WLAN) devices; c) a design which has been integrated with IEEE 802.1x [8], [9] to provide a system whereby clients and access points must query an authentication server.

The wired equivalent privacy (WEP) security scheme defined in the IEEE 802.11b standard has effectively been enhanced to an implementation known as temporal key integrity protocol (TKIP), designed for use in legacy systems and which was offered to the consumer by the interim wi-fi protected access (WPA) standard as IEEE 802.11i was being finalized. New 802.11 stations and access points are expected to implement a more secure and modern scheme described by IEEE 802.11i that is based on AES. There has been much fluctuation in the evolvement of the IEEE 802.11i standard. As such, two AES schemes have been proposed, of which the royalty-free, well-understood and proven cipher block chaining message authentication code (CBC-MAC) protocol (CCMP) appears to have emerged as dominant at the expense of the new, licensable and efficient wireless robust authentication protocol (WRAP) based on Rogaway's offset codebook mode (OCB) [10] for block ciphers.

Previous research into cryptographic microprocessor architectures has included extensions to instruction sets in order to increase the performance of symmetric key [11] and asymmetric key [12] cryptographic algorithms. Also, Fiskiran and Lee [13] have developed a data-scalable, general-purpose processor architecture with cryptographic extensions. However, these previous architectures do not utilise hardware coprocessors and are targeted at cryptographic algorithms rather than applications. Commercial WLAN security solutions do exist, such as Elliptic Semiconductor's CCM Internet protocol (IP) core [14] and Helion's 802.11i CCM IP core [15]. These efficient hardware cores integrate into application-specific integrated circuit (ASIC) designs, but only perform one WLAN protocol and do not offer the versatility of software. Cavium Network's NITROX processors [16] offer impressive data throughputs and versatility for numerous security applications, but are expensive in terms of area and thus, lack the compactness of a dedicated WLAN security solution. Also, the Cavium processors do not support the WEP or WRAP protocols, which are required to provide backwards compatibility.

In this paper, a dedicated WLAN security architecture is proposed. It comprises the authors' own primitive reduced instruction set computer (RISC) processor design and two hardware coprocessors, which perform AES and RC4 encryption. The RISC processor is designed to not only execute a standard range of arithmetic and logic instructions, but also dedicated cryptographic instructions that are required to implement WLAN protocols, such as Michael authentication [3], a packet authentication algorithm developed for IEEE 802.11i and 32-bit cyclic redundancy checks (CRC32). The WLAN processor has been designed specifically to perform the frame processing requirements of WEP, TKIP, WRAP, and CCMP as specified in Draft 3.0 of the IEEE 802.11i standard. It should be noted that WRAP was not included in the final IEEE 802.11i standard. The programmability of the processor also provides the ability to manipulate packet types, such as AES-CCM or AES-OCB encap-

Manuscript received April 3, 2004; revised February 15, 2005. This work was supported by Amphion Semiconductor Ltd., Belfast, U.K. This paper was recommended by Associate Editor S.-G. Chen.

N. Smyth is with Amphion Semiconductor Ltd., Belfast, Northern Ireland BT9 6SB, U.K., and Queen's University of Belfast, Belfast NT7 1NN, U.K. (e-mail: neil.smyth@conexant.com).

M. McLoone and J. V. McCanny are with The Institute of Electronics, Communications and Information Technology (ECIT), Queen's University of Belfast, Belfast NT7 1NN, U.K. (e-mail: m.mcloone@ecit.qub.ac.uk; j.mccanny@ecit.qub.ac.uk).

Digital Object Identifier 10.1109/TCSI.2006.877888

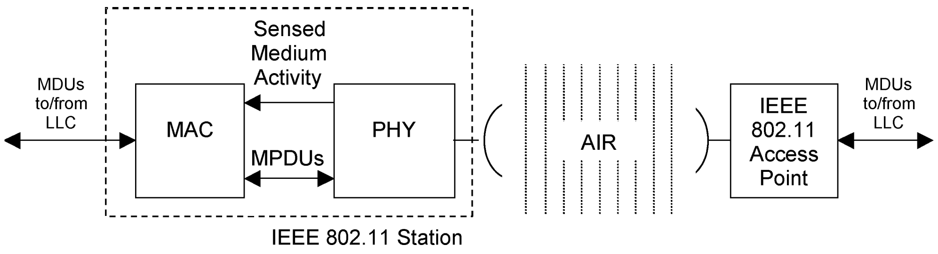

Fig. 1. IEEE 802.11 device-to-device interface.

sulation, which can provide functionality for IP security (IPSec) [17], [18].

The structure of the paper is as follows. In Section II, a brief background is given to the AES and RC4 algorithms and the basic operation of an IEEE 802.11 wireless network. Section III describes the operation and architecture of the processor and details how it generates IEEE 802.11 frames. The performance of the processor is outlined in Section IV. Finally, conclusions are given in Section V.

II. Background on AES, RC4, and IEEE 802.11

A. AES (Rijndael) Algorithm

AES [19] is a block cipher specified by the National Institute of Science and Technology (NIST) and is a standardized form of the Rijndael symmetric cipher. AES has a 128-bit block size and a variable key length of 128, 192 or 256 bits. This fast high security cipher is currently being introduced into many hardware and software security products, driven by the need for increased security in Internet traffic and many other multimedia products.

B. RC4 Stream Cipher

RC4 [4] is a proprietary stream cipher developed by Ron Rivest for RSA Data Security Inc. Stream ciphers are a class of symmetric-key encryption that are of particular interest to communications systems, such as WLAN, as they possess the important property of having no error propagation.

C. IEEE 802.11 Standard

The IEEE 802.11 standards define the MAC and Physical (PHY) layers of wireless LAN. The original standard described a wireless communication technology that operated at 1 megabits per second (Mbps). The IEEE 802.11b amendment introduced in 1999 increased the maximum throughput to 11 Mbps, while the newly created IEEE 802.11 a and g standards have introduced new technologies to increase the maximum theoretical throughput of this wireless communication technology to 54 Mbps.

As IEEE 802.11 is a form of wireless communication, it does not offer the inherent security of a wired LAN, as wireless communications disseminate information indiscriminately. To offer a level of security similar to that of wired LANs, the optional WEP amendment was introduced. This provides a means of confidentiality and authentication in the packetized data, through the use of the RC4 stream cipher for encryption and cyclic redundancy checks to provide a checksum for authentication purposes. WEP has been shown to be a weak security protocol [20] with many flaws [21], [22] and manufacturers have improved upon the standard by introducing their own amendments and enhancements.

To address the need for enhanced security as the uptake of wireless communications increases, IEEE developed the 802.11i standard for enhanced MAC security. This new standard provides a standardized upgrade of the WEP scheme for implementation on legacy systems, which is known as TKIP. However, new devices are expected to use the higher security AES block cipher, in either the CCMP or WRAP incarnations.

The basic outline of the processing layers in an IEEE 802.11 station is illustrated in Fig. 1. The MAC layer accepts data for transmission in the form of MAC data units (MDUs) from the logical link layer (LLC) in the system. The MAC creates and passes MAC physical data units (MPDUs) and other control and management packetized data (known as frames) to the PHY layer. The PHY performs modulation of the input frames to produce output data suitable for transmission over the wireless medium. By monitoring the activity on the wireless medium through the PHY, the MAC will determine that it can transmit data if the wireless medium is inactive.

Encryption and other cryptographic processing of frames in IEEE 802.11i occur at the MAC layer, prior to passing frames to the PHY. All frames delivered to the PHY from the MAC are composed of header fields, an optional data payload field and finally a frame check sequence (FCS) composed of a CRC32 checksum for error detection purposes. The security schemes in IEEE 802.11i only alter the data payload and subsequently the FCS field that is calculated over the data field. The header field may be used for creating parameters for authentication purposes.

III. WLAN Security Processor Architecture

The novel WLAN security processor proposed in this paper and depicted in Fig. 2, comprises the basic elements of any RISC processor—a memory controller to interface with RAM, an operation decode unit, a register bank, write-back logic, an arithmetic and logic unit (ALU) and a barrel shifter. In addition to these units, RC4 and AES encryption coprocessors have been added and are accompanied by IEEE 802.11i specific instructions to provide support for CRC32 checksums and Michael authentication tags.

The processor can execute the majority of instructions in two or three cycles. This allows for simple control logic and pipelining of execution logic, resulting in few logic levels between registers in the design. As a result, it is able to operate at the target frequency of 80 MHz in field-programmable gate

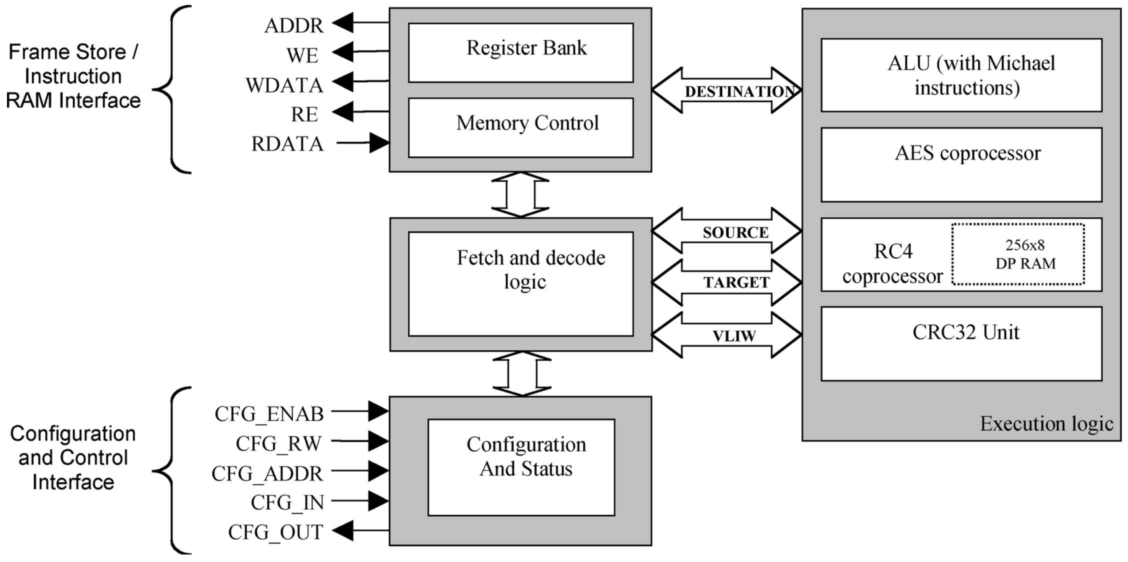

Fig. 2. WLAN security processor block diagram.

array (FPGA) technologies, a common operating frequency in commercial MAC/PHY products.

A. Design Rationale

The underlying design of the WLAN security processor is the authors' own simple RISC execution pipeline, which is capable of a standard range of arithmetic and logic instructions. This simplified processing element is enhanced to specifically accelerate and provide sufficient performance for WLAN security encapsulation.

This is achieved with the addition of specific instructions required to implement WLAN protocols, such as CRC generation and Michael authentication. These involve simple logical operations requiring bit shifting and XOR operations. Such operations could be performed using the basic ALU instructions of the processor. However, a large number of instructions would be required. As these are commonly used operations in the target application, hardware instructions with limited gate counts have been added to the instruction set of the WLAN processor.

Taking this enhancement of a commonly used operation a step further, complex encryption operations such as AES and RC4 may also be accelerated using a number of RISC instructions. For example, AES may use a number of instructions in order to implement a complete Rijndael round transform. However, the processor must implement numerous instructions in series to encrypt an entire block and each of these instructions, alongside any expanded AES keyspace, must be stored in RAM. In the WLAN processor design described here, greater performance is achieved using separate coprocessors which perform the operations in parallel with the main processing pipeline in a single instruction. In addition, keyspace generation is performed on-the-fly in hardware requiring no extra RAM storage.

The use of coprocessors to accelerate the complex encryption algorithms also allows AES or RC4 processing to be performed in parallel to other operations, such as data fetch/store to main memory. This, therefore, allows complex RC4 encryption to be performed in parallel to load/store and Michael authentication. This approach offers the higher throughput required for modern wireless applications, while maintaining a lower clock frequency that is important to reduce power dissipation. The coprocessors used in the architecture are based on commercially available RC4 and AES cores [23].

B. WLAN Security Processor Description

Synchronous read RAM is used to efficiently store the microcode that defines the frame encapsulation schemes, all input frames and all generated output frame data. Before processing can commence, the input data must be written into memory and the processor's configuration registers must contain the address pointers for the aperture of RAM containing both the input and output frames.

The WLAN processor has two distinct and simple interfaces, which are shown in Fig. 2. These include a simple 32-bit input–output (I/O) bus to interface with an external memory that contains the instruction and data memory and a second 32-bit I/O bus which interfaces to the processor's configuration interface. Both of these simplified busses provide a simple bridge to commonly used processor busses, such as the autoregressive model (ARM) advanced peripheral bus (APB) [24].

The processor has two external interfaces, as shown in Fig. 3. The first is for any controlling MAC and consists of a simple data bus allowing the MAC to access the memory-mapped registers of the WLAN security processor. The second interface is to external RAM storage and can be one of many different configurations. The lower address range of memory is utilized as instruction RAM. The size of this memory region is dependant on the instruction microcode required. The upper address range is used as a frame buffer. ROM may be used to store the instruction RAM when the size of this memory is fixed.

The most desirable memory configuration is to have a separate programmable ROM configured to store the microcode and a dual-port RAM to act as a frame buffer. This allows the microcode to be programmed once and avoids instructions being lost at power off. A dual-port frame buffer allows the MAC and

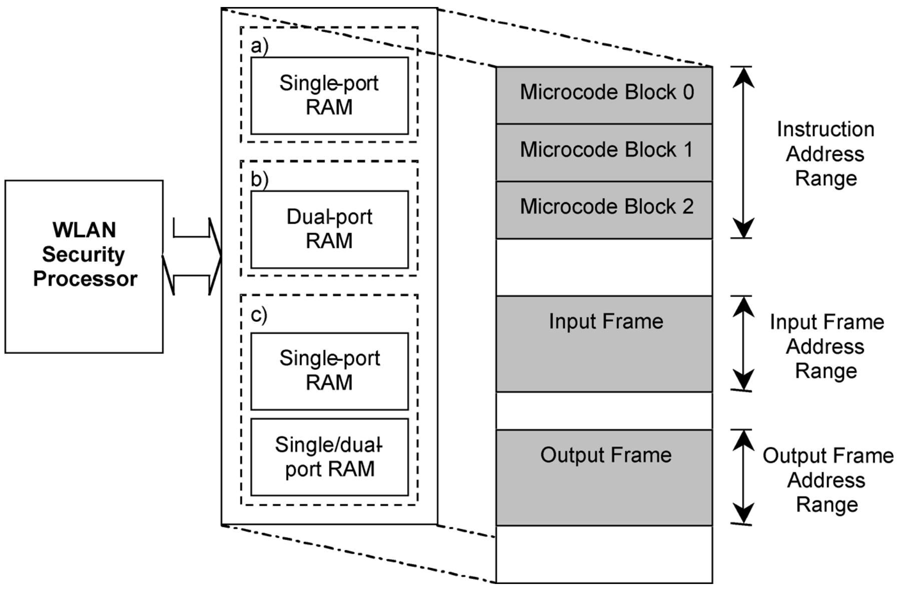

Fig. 3. RAM configurations.

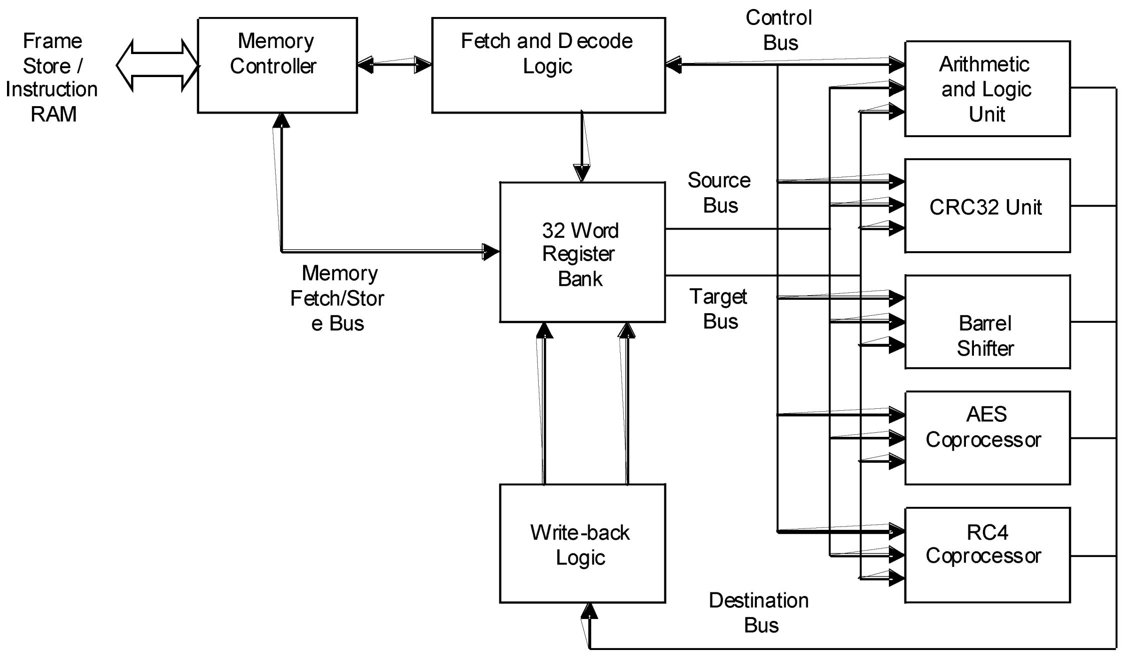

Fig. 4. Execution block diagram.

WLAN security processor to access data simultaneously. If a more compact RAM configuration is desirable, one single-port RAM can be used to store instructions and to act as a frame buffer, although this requires more careful access to RAM from the MAC and security processor.

The register bank is composed of a 32-word register file, utilizing two read ports and a write port. There are no register windows and the processor has access to all 32 registers. The 32-bit codeword is segmented into five sections, to provide: (a) an 8-bit instruction code; (b) a 5-bit source A; (c) a 5-bit source B; (d) a 5-bit destination register, and (e) a 9-bit region used to store miscellaneous data. The segmented regions of each instruction provide the instruction decode logic with all of the information required to determine the data to be manipulated, how to manipulate it and where to place the result.

When processing an instruction, the instruction codeword is first fetched from memory and passed to the decode logic. In the next clock cycle the 32-bit codeword is extended to create specific control data for the ALU, barrel shifter or coprocessors that are to be executed and to read source data from the register bank. In the second clock cycle the extended instruction data is passed to the relevant execution hardware logic alongside the input data from the register bank, where it is then manipulated. In the third and final cycle, the resulting data is written back to the register bank. The execution pipeline shown in Fig. 4 is highly simplified and requires three cycles to perform most instructions.

Control of the input and output address pointers is left to the rest of the system. In the event that the aperture in the RAM, selected by the address pointers, is overrun by the operating en-

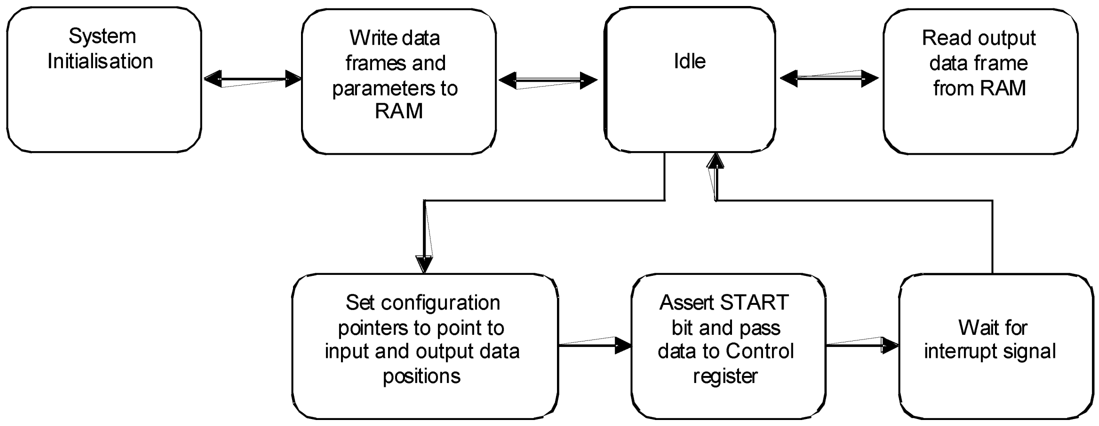
System Initialisation
Write data frames and parameters to RAM
Idle
Read output data frame from RAM
Set configuration pointers to point to input and output data positions
Assert START bit and pass data to Control register
Wait for interrupt signal

Fig. 5. State diagram describing host processor control.

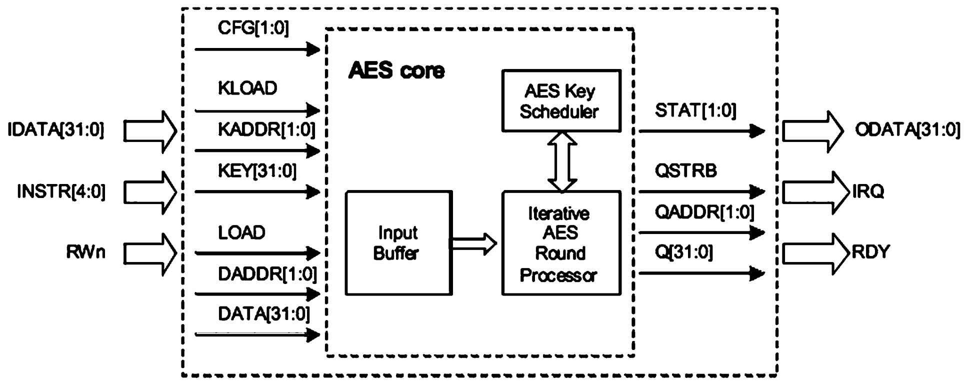

Fig. 6. AES coprocessor block diagram

capsulation program then an interrupt signal is asserted and all processing ceases until the host microprocessor re-initializes the security processor.

In terms of operation, the state diagram shown in Fig. 5 illustrates a basic overview of the steps a host microprocessor must take in controlling the security processor in an 802.11 MAC. Once the processor has been initialized with microcode and the frame data to be processed has been written to the frame store, the processor is ready to begin encapsulation. All that must be done by the host microprocessor to encapsulate the frame is to set the address pointers to indicate where the input and output frames are located in RAM and issue a start command to the processor. When encapsulation of a frame is complete, an interrupt is generated. The encapsulated frame can then be read from RAM by the MAC host microprocessor.

C. Cryptographic Coprocessors

1) AES: A commercial AES core [23] has been utilised in the creation of the proposed security processor design. This IP core is capable of both encryption and decryption and uses a fixed key length of 128-bits. It features a 32-bit datapath and an on-the-fly key scheduler that negates the need to store the expanded keyspace. The core has been interfaced with a coprocessor bus so that it can be controlled by the processor's execution pipeline. The output data is buffered such that it may be accessed as necessary. The status signals from the AES core have been manipulated to form an interrupt. The basic architecture of this core is outlined in Fig. 6.

2) RC4: The RC4 core [23] used interfaces to the execution pipeline through a coprocessor bus and is shown in Fig. 7. This RC4 core comprises a 256-byte dual-port RAM used to store the RC4 state array. Simple state machine and counter logic is used to manipulate the RAM contents in order to swap data bytes and perform permutations on the contents. The core must be initialized with a key, an operation that requires 1152 cycles. The RC4 core produces a stream of pseudorandom data that it XORs with an input byte stream.

D. Processing Encapsulated IEEE 802.11 Frames

IEEE 802.11i requires the use of two basic encryption primitives—AES and RC4. WEP and TKIP use RC4 for confidentiality purposes to transform plaintext to/from ciphertext. CRC32 and Michael are used to provide authentication in the form of message integrity check (MIC) and integrity check value (ICV) fields in TKIP and WEP frames respectively. These values are appended to the MPDU of a frame, while initialization vectors and miscellaneous control data are inserted at the start of the MPDU.

An RC4 coprocessor is used to perform all RC4 processing. Together with CRC32 support and other specialized instructions (such as support for the Feistel-based Michael authentication algorithm), the WLAN security processor can efficiently perform all 802.11 and 802.11i security encapsulation.

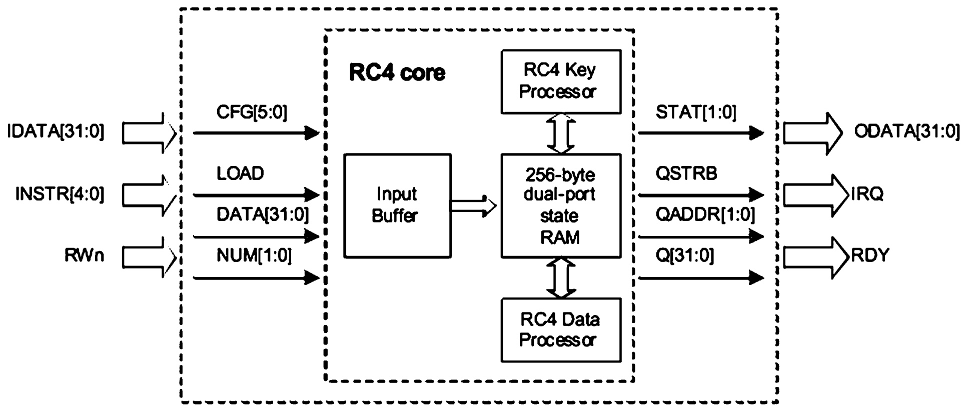

Fig. 7. RC4 block diagram.

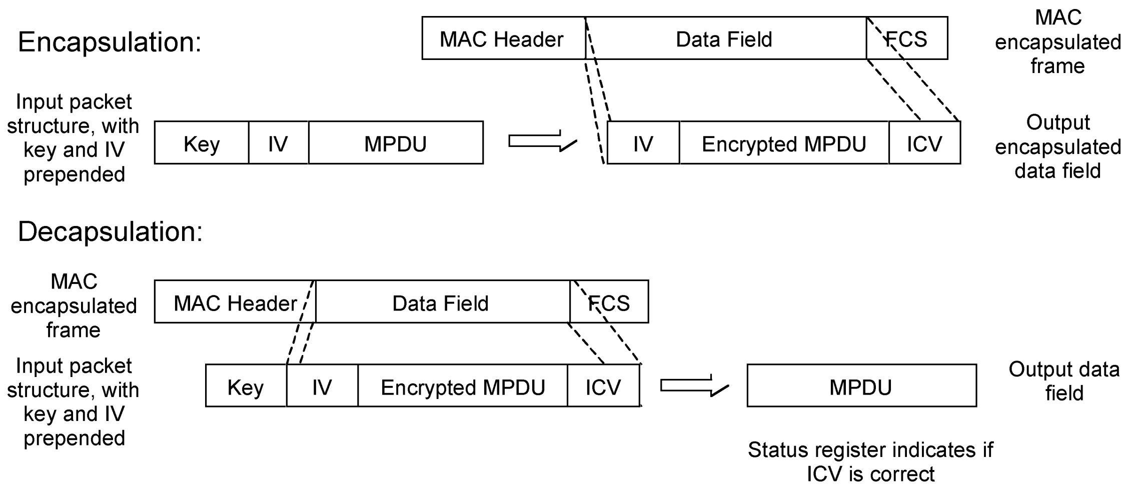

Fig. 8. WEP encapsulation and decapsulation frame structure.

WRAP and CCMP utilize AES to perform all confidentiality and authentication services. WRAP is based on AES-OCB and is more efficient than CCMP in terms of the AES processing required, but is a licensed scheme. CCMP is a royalty free scheme based on AES-CTR and AES-CBC-MAC and has a proven record of security in comparison to AES-OCB.

The WLAN security processor has an embedded AES core acting as a coprocessor, which provides AES functionality for 128-bit key sizes. In order to increase performance and flexibility, it is possible to perform AES and/or RC4 in parallel to other instructions in order to increase data throughput. The range of instructions provided enables the use of current and possibly many future block cipher modes of operation that could be applied to AES.

1) WEP Frame Processing: The following is an example of how the WLAN security processor implements WEP according to the developed reference software. The microcode that has been written to implement WEP on the WLAN security processor is provided in Appendix A. WEP is largely similar to TKIP and can reuse much of the same microcode. The first step is to write the input frame (known as an MPDU, or MAC protocol data unit) and its particular parameters to an aperture of RAM in a structure predefined by the embedded software, as shown in Fig. 8. The software will initialise the RC4 coprocessor with the relevant state array according to the key used.

If encapsulation is being performed, as indicated by the control register, the initialization vector (IV) is written to the output frame aperture. The length of the MPDU is contained in the control register and is checked when using the "IN" instruction to read words sequentially from the input aperture to the general purpose (GP) registers. These 32-bit words are then passed to the RC4 coprocessor for encryption and to perform a CRC32 generation instruction (the CRC32 checksum is reset with every new frame automatically). The encrypted data words are written sequentially to the output frame aperture using the "OUT" instruction, which allows data to be written to RAM sequentially as if being written to a first-in first-out (FIFO).

When the last 0–3 bytes of data have been encrypted and written back to GP registers, the final CRC32 generation instruction is executed and the CRC32 checksum is read from the coprocessor and written to GP registers. The checksum, known as an ICV and the last few encrypted bytes are aligned correctly in the register bank and can then be written to the output aperture. A software exception is then executed which generates an interrupt and returns the security processor to an idle state.

At this point, the MAC host microprocessor will have been informed that an encapsulated MPDU is contained in the output aperture. The MAC layer can then use this encapsulated MPDU to easily construct a MAC frame by pre-pending the relevant

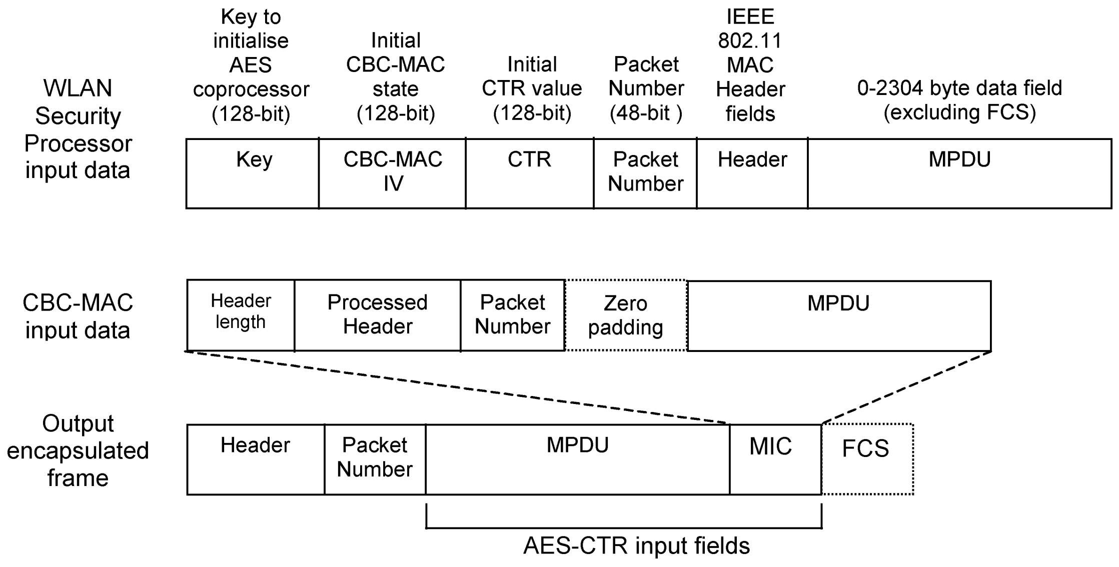
WLAN Security Processor input data
Key to initialise AES coprocessor (128-bit)
Initial CBC-MAC state (128-bit)
Initial CTR value (128-bit)
Packet Number (48-bit)
IEEE 802.11 MAC Header fields
0-2304 byte data field (excluding FCS)
Key
CBC-MAC IV
CTR
Packet Number
Header
MPDU
CBC-MAC input data
Header length
Processed Header
Packet Number
Zero padding
MPDU
Output encapsulated frame
Header
Packet Number
MPDU
MIC
FCS
AES-CTR input fields

Fig. 9. CCMP encapsulation frame structure.

IEEE 802.11 header and appending a frame check sequence (FCS) CRC32 checksum.

Decapsulation of WEP frames is very similar and reutilises some of the same microcode, as does the TKIP reference software. The major differences with TKIP are the use of Michael instead of CRC32 for authentication purposes and the use of an RC4 key stream that alters with every frame rather than one that remains static. The RC4 coprocessor is designed to be initialized for every new frame, which is a necessity for TKIP and requires 1152 cycles before it is ready to process data. During initialization the processor can continue to execute non-RC4 instructions.

2) CCMP Frame Processing: CCMP is less complex than WRAP. However, it places much more of a burden on the processing power of any software or hardware implementation as it requires an additional second AES pass for every 128-bit AES block. The use of a coprocessor in the WLAN security processor helps overcome this problem by allowing non-AES instructions to be performed while the AES coprocessor is busy.

The microcode that has been written to implement CCMP on the WLAN security processor is given as an example in Appendix B.

CCMP processing is performed in an identical manner to WEP, omitting RC4 for AES. The input data structure, as shown in Fig. 9, is arranged to aid the WLAN security processor such that extra cycles are not consumed to correctly align fields of the frame. The key occupies the first 4 words of the input data, which is directly read from RAM and written to the AES coprocessor. The CBC-MAC IV, the header length (encoded in a 32-bit field to simplify data alignment), the initial CTR value and the frame's Packet Number (PN) follow the key.

The header fields are first processed according to the parameters indicated in the control register to remove the duration field and mask the retry bit to zero, as these can change upon retransmission of a frame. This processed header is aligned with the header length (obtained from control register) and the PN. This data is then processed using the CBC-MAC algorithm, with zero padding of the last bytes if necessary. Simultaneous to processing the header and performing any CBC-MAC operations, the frame header is output to RAM.

Once the header has been fully processed, the data field can be operated on. Each 128-bit block of the MPDU is processed by interleaving the CBC-MAC and CTR algorithms, with the CTR blocks sequentially output to RAM. The last AES-CTR block of the MPDU is truncated according to the data length (stored in the control register) and the MIC generated by the CBC-MAC algorithm is appended for AES-CTR encryption. The WLAN security processor can process data using non-AES instructions while data is being encrypted, allowing blocks of data to be prepared in the register bank in parallel to AES encryption, thus improving the overall throughput. The CRC32 instructions may also be used to calculate the FCS field if desired.

The microcode required for CCMP is more complex than that required for any of the IEEE 802.11 security schemes. This is largely due to the requirement for header processing, the cycles consumed in aligning the different data fields and the two concurrently operating AES modes of operation (CTR and CBC-MAC).

E. Instruction Set

Table I provides a brief description of the cryptographic instructions utilised in the WEP and CCMP microcode given in Appendixes A and B.

IV. Performance Evaluation

The WLAN security processor proposed in this paper has been described in Verilog and modelled with a cycle-accurate C++ model, which identically matches the ports in each Verilog RTL module in the design. It was simulated using ModelTech ModelSim and the functionality of the core was verified against the functional C++ model using self-checking testbenches. Test vectors were obtained from various sources, such as the NIST [6], the IEEE 802.11 Task Group I [3] and the IETF [17], to

TABLE I
CRYPTOGRAPHIC INSTRUCTIONS

Instruction	*Description*
ARC4 INIT (*r*)	Initialise the RC4 coprocessor for operation with the number of bytes specified in register *r*.
ARC4 NUMB (*x*)	Set the number of bits being written to the RC4 coprocessor (*x* is equivalent to the number of bits, a value of 0 equals 32 bits).
ARC4 KEY (*r*)	Load the RC4 coprocessor with a 32-bit word containing key data.
ARC4 LASTKEY *r*	Load the RC4 coprocessor with the last word of key data and initiate setup of the RC4 state array.
ARC4 READ (*r*)	Read a 32-bit word of data from the RC4 coprocessor and place in the specified register.
ARC4 WRITE (*r*)	Write a 32-bit word from the specified register to the RC4 coprocessor, which automatically encrypts the data.
AESWRITE (*a, b*)	Load the AES coprocessor register *b* with the data contained in register *a*. Address *b* of 0-3 specifies the key buffer, 4-7 the encryption data buffer and 8-11 the decryption data buffer. Writing to address 7 or 11 initiates and AES operation.
AESREAD (*a, b*)	Read from coprocessor address *a* of 0-3 and write the encrypted/ decrypted result to register *b*.
CRC32 NUMB(*x*)	Set the bitlength of any input operand to the CRC32 logic to *x* bits (0 represents 32 bits).
CRC32 GEN (*r*)	Update the CRC32 checksum using the data contained in register *r*.
CRC32 PULL (*r*)	Read the CRC32 checksum and write to register *r*, resetting the CRC32 checksum for future use.

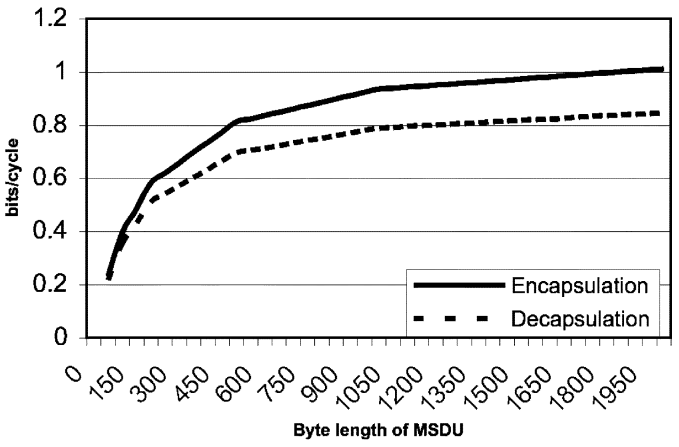

Fig. 10. WEP performance.

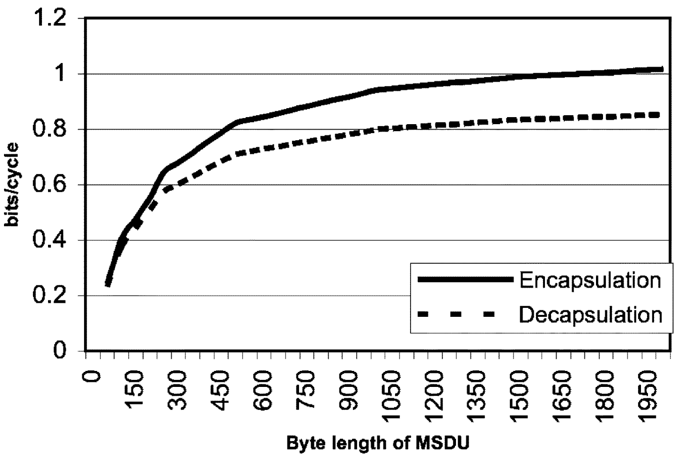

Fig. 11. TKIP performance.

verify the capability of the core to perform RC4 and AES encryption, and the various packet encapsulation schemes. The C++ model executable allows accurate debugging of microcode and fast performance evaluations to be made, such as the bits per cycle performance shown in Figs. 10–13. This allows microcode to be tested on a hardware model and cycle counts to process MPDUs to be rapidly collated.

RC4 requires an initialization period of 1152 cycles regardless of data payload length. Initialization of coprocessors and the register bank for every new packet or key change can be expected to have a greater effect upon small data fields. This is because the number of cycles required for initialization is fixed for each security scheme, hence contributing to larger performance degradation in smaller frames, particularly when RC4 is utilized. This is illustrated in Figs. 10 and 11.

AES-based CCMP requires less initialization than the RC4-based WEP or TKIP, as illustrated by Fig. 12. This is largely attributable to the absence of key initialization in AES. The degradation in performance of encryption to decryption in WEP, TKIP, and CCMP is attributable to the additional processing required in the decryption algorithms. WRAP encryption and decryption are largely identical and therefore have an identical

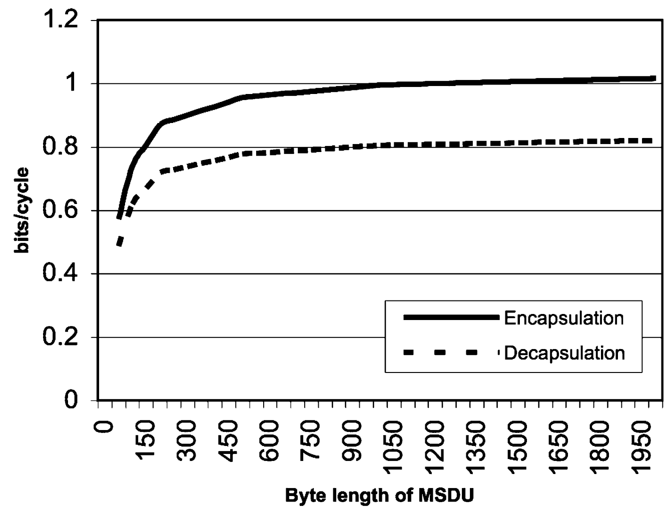

Fig. 12. CCMP performance.

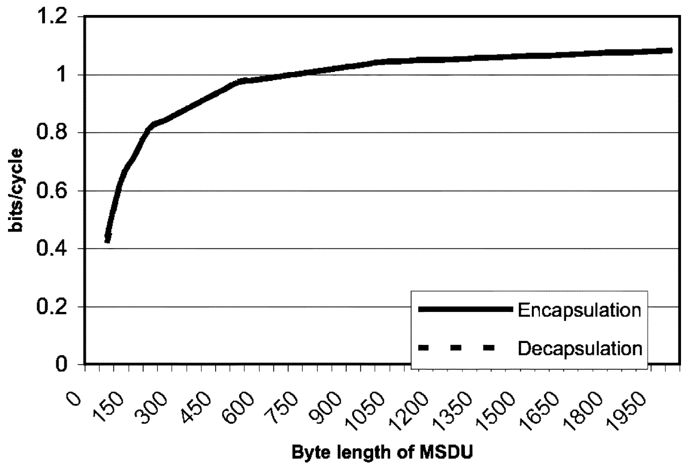

Fig. 13. WRAP performance.

processing bandwidth, as seen by the performance illustrated in Fig. 13.

WRAP has a 50% reduction in AES processing requirements compared to its AES-based counterpart CCMP. However, it also requires more complex initialization and data processing to be performed using the less efficient general-purpose instructions—although this can be performed in parallel to the AES encryption. Therefore, although 50% less AES processing is required per data block, the extra processing required to perform encryption and authentication in WRAP results in only 10% greater throughput when compared with CCMP.

The WLAN processor was synthesized using Synplify Pro to create netlists for FPGA implementation. The Verilog RTL includes compile-time parameters that implement technology dependant resources, such as RAM. Altera Quartus II and Xilinx Foundation Series 5.2 were used to perform place and routing of the netlist onto Altera Stratix and Xilinx Virtex II devices, respectively. Synopsys Design Compiler was also used to synthesize the core using TSMC 0.13-μm standard cell libraries under worst case conditions. The performance results obtained are illustrated in Table II.

The WLAN security processor is composed of three functional blocks, the processor control and execution logic, and the

TABLE II
WLAN SECURITY PROCESSOR TECHNOLOGY RESOURCE USAGE

Technology	*Logic Resources*	*Timing Constraint*	*RAM Resources*
TSMC 130 nm	60.4k gates	250 MHz	405.8k gates
Xilinx Virtex2 -5	3474 slices	80.3 MHz	15 BRAM
Altera Stratix	6873 LE	102.4 MHz	15 M4K

TABLE III
TSMC 0.13-μm LOGIC SIZE WITH 250-MHz CLOCK RATE

Block	*Logic Resources*	*RAM Resources*
Control and Execution logic	36.8k gates	32x32 3-port 4096x32 dual-port
AES coprocessor	17.6k gates	-
RC4 coprocessor	6k gates	256x8 dual-port

TABLE IV
PERFORMANCE COMPARISON OF COMMERCIALLY AVAILABLE SECURITY PROCESSORS

Solution	WEP	TKIP	WRAP	CCMP	Programmable	Max. throughput
WLAN Security Processor	Y	Y	Y	Y	Y	>275 Mbps
Elliptic Semiconductor [14]	Y	Y	N	Y	N	300 - 700 Mbps
Helion 802.11 CCM IP Core [15]	N	N	N	Y	Y	<2 Gbps
Cavium Networks Nitrox Processors [16]	N	Y	N	Y	Y	50-10000 Mbps

two encryption coprocessors (AES and RC4). The logic size of these blocks as implemented on TSMC 0.13-μm standard cell libraries under worst conditions is illustrated in Table III. The RAM size may differ depending upon application. The figures quoted are worst case (i.e., one 256×8 dual-port RAM for RC4 functionality, one 4096×32 dual-port RAM for packet buffer and instruction memory). The larger buffer RAM may be single-port, and may be reduced in size depending upon the required specification.

A comparison of the proposed WLAN processor with the commercially available solutions outlined in Section I is provided in Table IV.

The WLAN security processor offers a dedicated solution to wireless security, providing efficient hardware acceleration for the complex operations of encryption and a software driven execution pipeline providing versatility. From Table IV it is evident that this design compares only moderately with currently available solutions in terms of throughput. However, the major advantage of the processor is that it achieves this with additional functionality to perform the WLAN protocols required to provide backwards compatibility to existing networks as well as those required in future IEEE 802.11i compatible networks.

Helion offer an AES-CCM IP core solution with a throughput of up to 2 Gbps and Elliptic Semiconductor provide a CCMP core, which runs at 300 –700 Mbps and can support the addition of WEP and TKIP functionality. However, limited design details of these commercial solutions are available. Also, since the common operating frequency in commercial MAC/PHY products is 80 MHz and the IEEE 802.11a/g standards only require a throughput of 54 Mbps, these architectures are a very expensive solution for wireless applications and lack the compactness of a dedicated WLAN processor. The proposed processor's power efficiency is lower in comparison to the commercial solutions due to the supplemental hardware and execution time required to process software. However, when compared to general-purpose processor solutions, it will be more power efficient since it utilises hardware accelerators. Since the AES and RC4 encryption algorithms are accelerated in hardware in only a small number of instructions, the processor has the advantage of a greatly reduced software footprint. For example, the processor performs AES in just 6 instructions in comparison to approximately 640 instructions on a general-purpose processor [25]. These six instructions include 2 to load the 128-bit data block and 4 to write the output block to the register file. A status register is read to determine if AES has completed.

V. Conclusion

In this paper, a novel WLAN security processor is described which incorporates IEEE 802.11i specific instructions and AES and RC4 coprocessors. It has been recognized that there is a security processing gap in wireless devices, caused by the low power and relatively low processing capabilities of such devices and the demands of complex security protocols on microprocessor technologies.

Cryptographic instructions contained in the instruction set architecture (ISA) of microprocessor technologies can significantly improve the performance of security protocols operating on such microprocessors. Another method to increase throughput is to implement a hardware block to perform all secure packet processing for a particular application, or provide certain functionality such as AES encryption and map this into a processor's address space as a peripheral device or a coprocessor with fixed functionality. Industrially available microprocessor technologies utilizing such techniques include ARMs SecureCore [24] family, MIPS' SmartMIPS [26] or ARC [27].

The design described here combines both of these approaches to provide a processor designed specifically to perform efficient cryptographic processing of WLAN frames, with little intervention from the host microprocessor. As the host microprocessor is no longer burdened by performing bulk encryption and packet formatting of 802.11 frames, more processing power can be used to enhance and improve other services on a wireless handset. For example, the user interface may be more feature rich and responsive, there may be less lag experienced when using data services and dedicated hardware can perform cryptographic functions more efficiently than a general-purpose processor thus improving battery life.

Providing a software engine on which to execute the packet processing using dedicated cryptographic instructions allows changes to be made to the method of encapsulation, while maintaining the efficiency and high throughput of hardware encryption coprocessors. The current fluctuations in IEEE 802.11i standards can be overcome by implementing the WLAN security processor into a design, as it can be reprogrammed to accommodate any changes to packet structure or security scheme. As the AES and RC4 encryption algorithms are performed in hardware with a handful of instructions, the WLAN security processor also has the advantage of a greatly reduced software footprint.

In comparison to existing solutions, which have been targeted at specific WLAN protocols, the security processor described in this paper offers support for all WLAN protocols and thus, support for backwards compatibility and future upgrade ability as standards evolve. Moreover, it achieves this extra functionality at a throughput rate required by current 802.11a/g standards.

Licensable security schemes such as WRAP can be implemented through software patches, made optionally available to those wishing to purchase the license in order to use a more versatile and feature rich WLAN device. The processor may be programmed to perform encapsulation of other packet types, such as IPSec packets utilizing AES or RC4 based encryption. The ability to enable extra functionality by a simple software upgrade can be used as a distinguishing feature of a WLAN product in a competitive marketplace and offers the user a degree of future-proofing in their wireless LAN system.

Appendix A
WEP Microcode

1. Each instruction listed is converted to a 32-bit instruction by an assembler utility.
2. Code is based on IEEE 802.11 WEP Security Enhancement.

```
// 1st 2 words contain 40-bit WEP key
// 3rd word contains 32-bit IV
// Subsequent data contains MPDU/ICV if decapsulation
REG 1 6 SET // Set reg0x01 to 6
RC4 INIT 1 // Initialize RC4 processor
SETWAIT RC4 // Set interrupt to RC4
IN 4 2 // Load WEP key and IV
REG 2 3 SET
REG 8 1 SET
REG 7 6 MOV
REG 7 2 MSBMASK
REG 3 4 MOV
REG 3 8 MSBMASK
REG 3 0 ENDIAN
REG 3 7 OR
REG 4 8 LSBMASK
REG 9 8 SET
REG 4 9 LSHIFT
REG 5 8 MSBMASK
REG 5 0 ENDIAN
REG 4 5 OR
WAIT
```

```
RC4 KEY 3
WAIT
RC4 LASTKEY 4
WAIT

REG 12 0 MOV
REG 13 12 SET
REG 12 13 RSHIFT 30

// Encapsulation (26)
OUT 6 0 // output IV for encapsulated frames
REG 1 4 SET
REG 0 1 MTEQUAL 34 // If true – normal encapsulation
GOTO 41 99 // Do final operation

REG 12 10 EQUAL 26
REG 12 13 LSHIFT
REG 0 12 XOR
REG 0 1 SUB 72

// Normal word encapsulation (34)
IN 4 0
RC4 WRITE 4
CRC32 GEN 4
REG 0 1 SUB
WAIT
RC4 READ 4
OUT 4 0 28

// Pad final bytes, pull out CRC32 ICV (41)
REG 0 10 EQUAL 67
IN 4 0
REG 4 0 MSBMASK
RC4 NUMB 0
RC4 WRITE 4
WAIT
CRC32 NUMB 0
CRC32 GEN 4
RC4 NUMB 10
RC4 READ 4
CRC32 PULL 5

RC4 WRITE 5
WAIT
RC4 READ 5

REG 6 5 MOV
REG 8 0 MOV
REG 0 10 EQUAL 61
REG 5 9 ROTR
REG 6 9 ROTR
REG 0 0 DECR 57

REG 5 8 LSBMASK
REG 1 8 SUB
REG 6 8 MSBMASK
REG 5 4 OR
OUT 5 1
RETURN

CRC32 PULL 5
RC4 WRITE 5
WAIT
RC4 READ 5
OUT 5 0 66

// Decapsulation (72)
REG 1 4 SET
REG 0 1 MTEQUAL 76 // If true – normal encapsulation
GOTO 83 // Do final operation
STATUS 0

// Normal word decapsulation (76)
IN 4 0
RC4 WRITE 4
REG 0 1 SUB
WAIT
RC4 READ 4
CRC32 GEN 4
OUT 4 0 73

// Pad final bytes, pull out CRC32 ICV (83)
IN 4 1
REG 2 4 SET
REG 2 0 ADD
RC4 WRITE 4
REG 2 1 SUB
WAIT
RC4 READ 4
REG 2 10 EQUAL 95
RC4 NUMB 2
RC4 WRITE 5
WAIT
RC4 READ 5 100

CRC32 PULL 5
REG 4 5 EQUAL 98
REG 4 1 SET 99
REG 4 0 SET
REG 1 0 CLR 113

CRC32 NUMB 2
CRC32 GEN 4
CRC32 NUMB 10
CRC32 PULL 6
REG 3 4 MOV
REG 4 0 LSBMASK
REG 3 0 MSBMASK
OUT 3 0

REG 4 5 OR
REG 4 9 ROTL
REG 2 0 DECR
REG 2 10 MORETHAN 109
REG 5 6 MOV 96

REG 2 0 CLR
REG 3 0 CLR
REG 4 0 CLR
REG 5 0 CLR
REG 6 0 CLR
REG 7 0 CLR
REG 8 0 CLR
REG 9 0 CLR
REG 12 0 CLR
REG 13 0 CLR
STATUS 4
```

APPENDIX B
CCMP MICROCODE

1. Each instruction listed is converted to a 32-bit instruction by an assembler utility.
2. Code is based on IEEE 802.11i draft 3.0.

```
IN 4 3 // Load in AES key
AESWRITE 4 0
AESWRITE 5 1
AESWRITE 6 2
AESWRITE 7 3 // Write AES key to AES core
REG 1 5 SET
REG 3 4 SET // register 3 set to 4
REG 1 3 LSHIFT
REG 2 9 SET
REG 1 2 OR // 0x59 flags field generated
IN 4 1 // Load A2 field into register 4 and 5
IN 24 1 // Load PN field into register 0x18 and 0x19
REG 2 2 SET
REG 4 2 LSBMASK
REG 1 0 ENDIAN
REG 4 1 OR
STATUS 0

// CCMP transmission notes:
//
// Input frame to be CCMP encapsulated has following format:
// 4 word KEY, 2 word A2, 2 word PN, 1 word CTR, variable length header, variable
// length data HLEN and DLEN is stored in cc_sec config register
//
// 1. 4 word key is loaded and passed to AES core
// 2. CBC-MAC IV in 16–19 is constructed from first 10 words
// 3. HLEN is stored in register 20 (decremented by 2)
// 4. CTR is stored in 21–24
// 5. PN is stored in register 25 –26

// CCMP Initialisation code
// NOTES: 1. will write 3 LSW of CBC-MAC to AES
// 2. will write input data in registers 4–7
// 3. will write CBC-MAC IV to 21–24
IN 4 3 // Load in AES key
AESWRITE 4 0
AESWRITE 5 1
AESWRITE 6 2
AESWRITE 7 3 // Write AES key to AES core
REG 1 5 SET
REG 3 4 SET // register 3 set to 4
REG 1 3 LSHIFT
REG 2 9 SET
REG 1 2 OR // 0x59 flags field generated
IN 4 1 // Load A2 field into register 4 and 5
IN 25 1 // Load PN field into register 0x1a and 0x1b
REG 2 15 SET
REG 8 15 SET
REG 2 3 LSHIFT
REG 2 8 OR // Set register 2 to mask 0x000000ff
REG 3 8 SET
REG 8 2 MOV
REG 8 3 LSHIFT
REG 2 8 OR // Set register 2 to mask 0x0000ffff
REG 4 2 AND // AND register 4 to mask out top 16 bits
REG 1 3 LSHIFT
REG 1 3 LSHIFT
REG 1 3 LSHIFT
REG 4 1 OR // Create MSW of CCM IB
REG 6 25 MOV
REG 9 1 SET
REG 3 9 LSHIFT
REG 6 3 LSHIFT
REG 28 26 MOV
REG 28 3 RSHIFT
REG 6 28 OR // 3nd word of CCM IB in register 6

REG 7 26 MOV
REG 7 3 LSHIFT // Place lower 2 bytes of PN in register 7
REG 28 0 MOV
REG 28 2 AND // Mask off DLEN
REG 7 28 OR // Create LSW of CCM IB -> 4–7 now contains CCM IB

IN 24 0
REG 28 7 MOV
REG 24 2 AND
REG 21 4 MOV
REG 21 2 AND
REG 2 0 NOT
REG 28 2 AND
REG 24 28 OR
REG 22 5 MOV
REG 23 6 MOV
REG 0 3 RSHIFT
REG 3 9 RSHIFT
REG 9 3 ROTR
REG 21 9 OR // create CTR IV -> 21–24
REG 2 0 NOT
REG 25 2 AND
REG 1 12 SET
SETWAIT AESRDY .INITCCMP
.CCMP_MAINLOOP_CHECK
REG 0 1 LESSTHAN .CCMP_BLOCK_PAD // do final CCMP processing
GOTO .CCMP_LOOP . CCMP_MAINLOOP_CHECK

// CCMP Loop code – 21–24
// NOTES: 1. expects MSW of CBC-MAC to be written to AES
// 2. expects input data in 4–7
// 3. expects CBC-MAC IV in 21–24
// 4. expects CTR IV in 12–15
// 5. expects 16 bytes for processing
.CCMP_LOOP
AESWRITE 24 7 // start CBC-MAC processing
AESREAD 0 8 // read AES-CTR output to 8–11
AESREAD 1 9
AESREAD 2 10
AESREAD 3 11
REG 15 0 INCR // increment CTR
REG 8 4 XOR // XOR plaintext with CTR output
REG 9 5 XOR
REG 10 6 XOR
REG 11 7 XOR
OUT 8 3 // output ciphertext
AESWRITE 12 4 // write CTR to AES
AESWRITE 13 5
AESWRITE 14 6
WAIT
AESWRITE 15 7
AESREAD 0 21 // read new CBC-MAC value
AESREAD 1 22
AESREAD 2 23
AESREAD 3 24
REG 21 4 XOR // XOR plaintext with current CBC-MAC in 21–24
REG 22 5 XOR
REG 23 6 XOR
REG 24 7 XOR
IN 4 3
GOTO .ENDIANCONV .WRITECBCMAC // do endian conversion
.WRITECBCMAC
AESWRITE 21 4 // write CBC-MAC plaintext to AES
AESWRITE 22 5
AESWRITE 23 6
REG 0 1 SUB // decrement data length counter by 16 (bytes)
RETURN // return to higher code level

IDLE (2)

// CCMP block endian conversion
// NOTES: 1. converts registers 4–7 to bigendian
.ENDIANCONV
REG 4 0 ENDIAN
REG 5 0 ENDIAN
REG 6 0 ENDIAN
REG 7 0 ENDIAN
RETURN

// Initial CCMP data processing
// NOTES: 1. expects data length in 0
// 2. expects input data space reserved in 4–7
// 3. expects CBC-MAC IV in 21–24
// 4. expects CTR IV in 12–15
// 5. expects more than 16 bytes of data
.INITCCMP
IN 4 3
GOTO .ENDIANCONV .CONT_INITCCMP // do endian conversion
.CONT_INITCCMP
REG 21 4 XOR
REG 22 5 XOR
REG 23 6 XOR
REG 24 7 XOR
AESWRITE 21 4
AESWRITE 22 5
AESWRITE 23 6
.CHECK_INITCCMP
REG 0 1 LESSTHAN .CCMP_BLOCK_PAD
GOTO .CCMP_LOOP .CHECK_INITCCMP
RETURN // return to higher level for final block processing and MIC generation
// CCMP AES block padder
// NOTES: 1. expects data length in 0
// 2. places data in registers 4–7
// 3. converts to bigendian
// 4. inserts zero-padding
.CCMP_BLOCK_PAD
REG 2 4 SET
REG 5 0 CLR
REG 6 0 CLR
REG 7 0 CLR
REG 0 5 MORETHAN .CCMP_DO_PAD
IDLE // should obtain MIC otherwise, i.e., GOTO command
.CCMP_DO_PAD
IN 4 0
REG 0 2 LESSTHAN .CCMP_PAD_LTA
REG 4 0 ENDIAN
REG 4 2 SUB .CCMP_RESUME_PADA
.CCMP_PAD_LTA
REG 4 0 ZPAD
REG 4 0 ENDIAN
GOTO .CCMP_FINAL 0 // goto write AES code
.CCMP_RESUME_PADA
IN 5 0
REG 0 2 LESSTHAN .CCMP_PAD_LTB
REG 5 0 ENDIAN
REG 5 2 SUB .CCMP_RESUME_PADB
.CCMP_PAD_LTB
REG 5 0 ZPAD
REG 5 0 ENDIAN
GOTO .CCMP_FINAL 0 // goto write AES code
.CCMP_RESUME_PADB
IN 6 0
REG 0 2 LESSTHAN .CCMP_PAD_LTC
REG 6 0 ENDIAN
REG 6 2 SUB .CCMP_RESUME_PADC
.CCMP_PAD_LTC
REG 6 0 ZPAD
REG 6 0 ENDIAN
GOTO .CCMP_FINAL 0 // goto write AES code
.CCMP_RESUME_PADC
IN 7 0
REG 0 2 LESSTHAN .CCMP_PAD_LTD
REG 7 0 ENDIAN
REG 7 2 SUB . CCMP_FINAL
.CCMP_PAD_LTD
REG 7 0 ZPAD
REG 7 0 ENDIAN
// CCMP Final processing code
// NOTES: 1. expects final padded data block in 4–7
// 2. expects CBC-MAC IV in 21–24
// 3. expects CTR IV in 12–15
.CCMP_FINAL
AESWRITE 24 7 // start CBC-MAC processing
AESREAD 0 8 // read AES-CTR output to 8–11
AESREAD 1 9
AESREAD 2 10
AESREAD 3 11
REG 8 4 XOR // XOR plaintext with CTR output
REG 9 5 XOR
REG 10 6 XOR
REG 11 7 XOR
OUT 8 3 // output last ciphertext block
WAIT
AESREAD 0 21 // read new CBC-MAC value
AESREAD 1 22
AESREAD 2 23
AESREAD 3 24
REG 21 4 XOR // XOR plaintext with current CBC-MAC in 21–24
REG 22 5 XOR
REG 23 6 XOR
REG 24 7 XOR
AESWRITE 21 4
AESWRITE 22 5
AESWRITE 23 6
AESWRITE 24 7 // start CBC-MAC processing
WAIT
AESREAD 0 21 // read new CBC-MAC value
AESREAD 1 22
AESREAD 2 23
AESREAD 3 24
OUT 21 1 // output MIC
IDLE
```

REFERENCES

[1] S. Ravi, A. Raghunathan, and M. Sankaradass, "Securing wireless data: System architecture challenges," in *Proc. ISSS'02*, Kyoto, Japan, Oct. 2002.

[2] S. Gayal and S. A. Vetha Manickam, *Wireless LAN Security Today and Tomorrow*. Pune, India: Center for Information and Network Security, Pune University, 2002.

[3] IEEE 802.11 Working Group, Task Group I IEEE 802.11 Wireless LAN Standards, 2004 [Online]. Available: http://grouper.ieee.org/groups/802/11

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. New York: Wiley, 1996, pp. 397–398.

[5] Original posting of RC4 Algorithm to Cypherpunks mailing list 2006 [Online]. Available: http:// cypherpunks.venona.com/archive/1994/09/msg00304.html

[6] AES (Rijndael) Specification and Information National Institute of Standards and Technology, Boulder, CO, 2004 [Online]. Available: http://csrc.nist.gov/encryption/aes/rijndael

[7] M. Welschenbach, *Cryptography in C and C++*. Berkeley, CA: Apress, 2001.

[8] J. S. Park and D. Dicoi, WLAN Security: Current and Future IEEE Internet Computing, Piscataway, NJ, (2003) [Online]. Available: http://computer.org/internet/

[9] J. Williams, Providing for Wireless LAN Security IT Professional, IEEE Internet Computing, Piscataway, NJ, (2002/2003) [Online]. Available: http://computer.org/internet/, 2

[10] P. Rogaway, M. Bellare, J. Black, and T. Krovetz, OCB: A Block Cipher Mode of Operation for Efficient Authenticated Encryption UC Davis, Davis, CA, (2004) [Online]. Available: http://www.cs.ucdavis.edu/ rogaway/ocb/

[11] J. Burke, J. McDonald, and T. Austin, "Architectural support for fast symmetric-key cryptography," in *Proc. Int. Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Nov. 2000, pp. 178–189.

[12] J. Groszschaedl and G. A. Kamendje, "Instruction set extension for fast elliptic curve cryptography over binary finite fields $GF(2m)$," in *Proc. IEEE Int. Conf. Application-Specific Systems, Architectures and Processors (ASAP)*, Jun. 2003, pp. 442–455.

[13] A. Murat Fiskiran and R. B. Lee, , N. Nedjah and L. de Macedo Mourelle, Eds., *PAX: A Datapath-Scalable Minimalist Cryptographic Processor for Mobile Environments, Embedded Cryptographic Hardware: Design and Security*. New York: Nova Science, 2004.

[14] Cryptography IP Cores Elliptic Semiconductor, Kanata, ON, Canada, (2006) [Online]. Available: http://www.ellipticsemi.com

[15] CCM IP Core Helion Technology, Cambridge, U.K., (2004) [Online]. Available: http://www.heliontech.com

[16] NITROX Processors Cavium Networks , Mountain View, CA, Jan. (2004) [Online]. Available: http://www.cavium.com

[17] Requests for Comments and Internet Drafts Internet Engineering Task Force (IETF) IPSec, (2004) [Online]. Available: http://www.ietf.org

[18] M. McLoone and J. V. McCanny, "A single-chip IPSec cryptographic processor," in *Proc. IEEE Workshop on Signal Processing Systems Design and Implementation—SiPS'02*, Oct. 2002, pp. 133–138.

[19] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*. New York: Springer-Verlag, 2002.

[20] J. A. LaRosa, WPA: A Key Step Forward in Enterprise-Class Wireless LAN (WLAN) Security Meetinghouse Data Communications, (2004) [Online]. Available: http://www.meetinghouse-data.com/landing/wp.shtml

[21] N. Borisov, I. Goldberg, and D. Wagner, Security of the WEP Algorithm (2005) [Online]. Available: http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html

[22] J. Walker, Unsafe at Any Key Size; An Analysis of the WEP Encapsulation IEEE 802.11 Committee, 2000.

[23] Cryptography IP Cores Amphion Semiconductor Ltd., (2004) [Online]. Available: http://www.amphion.com

[24] ARM SecureCore, Jan. (2004) [Online]. Available: http://www.arm.com

[25] K. Atasu, L. Breveglieri, and M. Macchetti, "Efficient AES implementations for ARM based platforms," in *ACM Symposium on Applied Computing*, 2004.

[26] SmartMIPS, (2004, Jan.) [Online]. Available: http://www.mips.com

[27] Accelerating Network Processing with Extensions to the User-Customizable ARC Tangenttm Microprocessor ARC International, (2004) [Online]. Available: http://www.arc.com/downloads/downloads-white-papers.html

Neil Smyth (M'03) received the Master of Engineering degree (with first class honors) in electrical and electronic engineering from Queen's University of Belfast, Belfast, U.K., in 2001. Since 2003, he has been working toward the Ph.D. degree (part time) at the same university.

He is currently a Senior Engineer at Amphion Semiconductor Ltd. (a subsidiary of Conexant Systems Inc.), Belfast, U.K., where he has been involved in the development of the company's cryptography, wireless local area network and video compression technologies. His research interests include cryptography system-on-chip implementation, including general-purpose cryptographic processing architectures and application-specific hardware acceleration.

Mr. Smyth is a member of the Institution for Engineering and Technology, U.K.

Máire McLoone (M'04) received the Master of Engineering degree (with distinction) in electrical and electronic engineering (with the first place) and the Ph.D.degree in digital signal processing from Queen's University of Belfast, Belfast, U.K., in 1999 and 2002, respectively.

She is currently in her third year of a 5-year Royal Academy of Engineering research fellowship conducting research into cryptographic algorithms and architectures for system-on-chip. She has authored a research book *System-on-Chip Architectures and Implementations for Private-Key Data Encryption* (Kluwer, 2003) and has over 30 peer-reviewed conference and journal publications. Her research interests include generic silicon architectures for symmetric and asymmetric cryptographic algorithms, security for wireless and ad hoc networks, hardware/software cryptographic system-on-chip architectures, and cryptography for constrained environments.

Dr. McLoone was presented with a U.K. Science Engineering and Technology (SET) Student of the Year award for best electronic engineering student in 1999. In 2000, she received the Beijing Institute of Technology (BIT) International Outstanding Student of the Year award. In 2004, she was presented with the Vodafone award for her work in high-speed data security at the Britain's Younger Engineers event held at the House of Commons, London, U.K.

She is a member of the Institution for Engineering and Technology, U.K. and International Association for Cryptologic Research.

John V. McCanny (M'86–SM'95–F'99) received the Bachelor's degree in physics from the University of Manchester, Manchester, U.K., the Ph.D.degree in physics from the University of Ulster, Ulster, U.K., and the D.Sc. (higher doctorate) degree in electrical and electronics engineering from Queen's University of Belfast, Belfast, U.K., in 1973, 1978, and 1998, respectively.

He is an international authority in the design of silicon integrated circuits for digital signal processing; having made many pioneering contributions to this field. He has co-founded two successful high technology companies, Audio Processing Technology Ltd., and Amphion Semiconductor Ltd. a leading supplier of SoC cores for video compression. He is currently Director of the Institute of Electronics, Communications and Information Technology at Queen's University of Belfast. He has published 250 major journal and conference papers, holds 25 patents and has published five research books.

Prof. McCanny is a Fellow of the Royal Society, the Royal Academy of Engineering, Institution for Engineering and Technology, U.K., and the Institute of Physics. He has won numerous awards, including a Royal Academy of Engineering Silver Medal for outstanding contributions to U.K. engineering leading to commercial exploitation (1996), an IEEE Third Millennium medal and the Royal Dublin Society/Irish Times Boyle Medal (2003). In 2002, he was awarded a CBE for his contributions to engineering and higher education.