Reprint

Hardware implementation of the Wired LAN Equivalent Privacy in 802.11 Wireless LANs

M. Iliopoulos and T. Antonakopoulos

The 3rd IMACS-IEEE International Multi-conference on Circuits, Systems, Communications and Computers – CSCC'99

ATHENS, JULY 1999

Hardware implementation of the Wired LAN Equivalent Privacy (WEP) in 802.11 Wireless LAN

M. Iliopoulos[1] and T. Antonakopoulos[2]

[1] Atmel-Hellas S.A., Multimedia & Communications Group,
Patras Science Park, Stadiou Street, Platani 26500, Patras.
[2] Department of Electrical Engineering and Computers Technology,
University of Patras, Rio 26500, Patras.
GREECE
e-mail: antonako@ee.upatras.gr

Abstract - As wireless LANs become more and more popular, data confidentiality is getting more and more critical. The IEEE802.11 standard for Wireless LANs (WLANs), which is a very significant milestone in the development of the wireless technology, embodies a ciphering/deciphering algorithm to offer wired LAN equivalent privacy to 802.11 LAN users. The ciphering/deciphering algorithm used in the IEEE802.11 standard is based on the RC4 pseudo-random generator algorithm developed by RSA Data Security Inc. This paper presents a hardware implementation of this ciphering/deciphering algorithm and describes how this implementation is integrated in the IEEE802.11 MAC logic.

Key-Words - WEP, RC4, wireless encryption, ciphering, IEEE802.11, ARM

1 Introduction

Wireless Local Area Networks (WLANs) are used in a variety of applications and substitute traditional LANs by providing mobility to their users. All traditional LANs were developed on the assumption that only users attached to the same wired transmission medium could exchange MAC frames. This is a type of privacy, since only authorized users could be physically attached to the network, and thus only these users could have access to the exchanged information. This assumption is not valid when WLANs are used, since an external to the network user could 'listen' on the unbounded transmission medium and collect unauthorized information. In order to solve this problem and to provide privacy equivalent to the wired LANs users, the IEEE802.11 standard defined the Wired Equivalent Privacy algorithm [1].

The Wired Equivalent Privacy (WEP) uses a cryptographic algorithm, or cipher, to encipher or decipher the user data [2]. Modern cryptographic algorithms, also called encryption algorithms (denoted by E), use a key sequence (K) to modify the plaintext data (P) to produce enciphered data (C):

$$E_K(P) = C$$

The reverse process, called decryption function (D), operates on the C data block to recreate the initial data block, P:

$$D_K(C) = P$$

A typical cryptographic algorithm can be considered as an electronic book in which a block of plaintext is bitwise XORed with a pseudo random key sequence. In the IEEE802.11 standard, the WEP algorithm, which uses such a pseudo random sequence, uses also an integrity check value to protect against unauthorized data modification. This process is based on CRC-32 calculations of the initial plaintext.

A general block diagram showing the E and D processes is illustrated in Figure 1. As it is shown in this figure, the plaintext sequence is XORed with the pseudo random number produced in the encryption function during the encryption phase and is sent over the wireless medium. At the end of the data transmission, the integrity value is XORed with the next pseudo random value and the enciphered integrity value is also sent over the air. At the receiver side, the decryption unit uses the same pseudo random sequence generator to XOR the

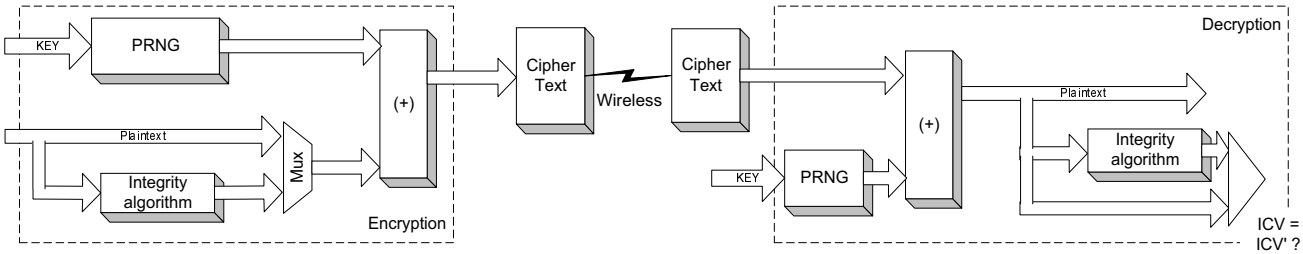

Fig. 1 *Block diagram of the Encryption/Decryption process*

received data in order to generate the initial plaintext and then passes the decrypted data to the CRC check engine. At the end of the packet reception, the received integrity check value is compared to the calculated CRC value. If they are the same the frame is forwarded to the upper layer, otherwise the frame is rejected.

2 The WEP Algorithm

The IEEE802.11 WEP algorithm uses the RC4 pseudo random generator algorithm [2] as the encryption function and an Integrity Check Algorithm (ICV) to protect from unauthorized data modification. When a packet is encrypted using the WEP algorithm the transmitted MPDU is expanded by eight bytes. Four bytes are added at the beginning of the PDU, and other four bytes are used at the end. The first three bytes of the IV field contain a part of the initialization vector (key) used by RC4, while the fourth byte indexes to one of the four 40-bit secret keys maintained by the wireless station. The format of the expanded PDU is shown in Figure 2.

2.1 The RC4 algorithm

The cryptographic algorithm used in WEP is the RC4 pseudo random generator algorithm from RSA Data Security Inc. This algorithm uses a 256-byte array (S), as the electronic cipher book, and two pointers i and j in order to produce the random data. At the beginning, the algorithm initializes each position of the array with the values 0 to 255. Then, the algorithm performs 256 random swaps (from 0 to 255), based on the value of the cipher key (K). The swaps are performed according to the following procedure:

$$S[i] \leftrightarrow S[j]$$

where:

$$j=(j+S[i]+K[i]) \bmod 256$$

The symbol $\leftrightarrow$ is used to denote exchange of content. These two steps constitute the initialization phase. At the next step, a random number is produced at the S position that comes from S[i]+S[j], where:

$$i=(i+1) \bmod 256,$$

$$\text{and } j=(j+S[i]) \bmod 256,$$

and at the same time the following swap is performed:

$$S[i] \leftrightarrow S[j]$$

The random number is then XORed with the plaintext data to produce the ciphered data. Because of the symmetry of the algorithm the same steps are applied in order to decrypt the data.

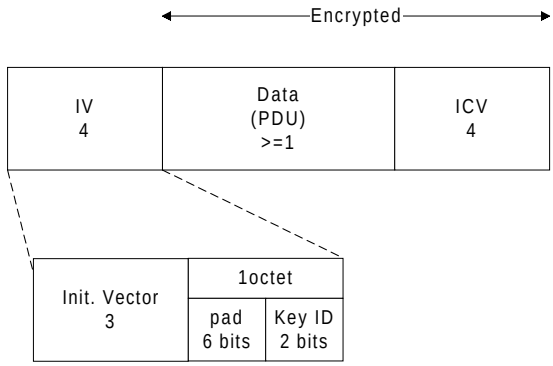

Fig. 2 *The expanded IEEE802.11 MAC-PDU*

2.2 The ICV algorithm

The ICV is a 32-bit field containing the 32-bit Cyclic Redundancy Check (CRC). This field is calculated over the plaintext bytes of the MAC PDU. This means that the ICV calculation happens before XORing the data during transmission and after XORing the data during reception. The ICV calculates the CRC according to the following polynomial generator of 32-degree:

$$G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8} +x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

The four bytes that remain after encrypting all the plaintext are appended at the end of MPDU, and are considered as MPDU data.

Keeping in mind that the RC4 algorithm produces a random byte in each cycle, the ICV algorithm should also be capable to calculate the CRC on 8-bit quantities, hence an 8-bit parallel implementation of the CRC-32 may be used.

3 WEP Implementation in hardware

The Wireless Equivalent Privacy (WEP) module implements the RC4 algorithm for data encryption/ decryption. The WEP module is programmed to encrypt/decrypt a block of data in the memory automatically without loading the processor. A detailed diagram of the module that implements the RC4 algorithm is illustrated in Figure 3. The basic blocks of the WEP module are:

a) *the SBOX array*, which stores the random numbers, and is a 256 byte RAM,
b) *the algorithm state machine* which implements the algorithm, produces the pseudo-random sequence and controls the read/write operations of data from/to the system RAM, and
c) *the CRC32 8-bit parallel engine*, which implements the ICV algorithm.

The state machine which controls the WEP module is illustrated in Figure 4. This state machine has five different stages as explained below, the *idle* state, the *initialize* state, the *RandomSwap* state, the *RandomGen* state and the *ICVCalc* state.

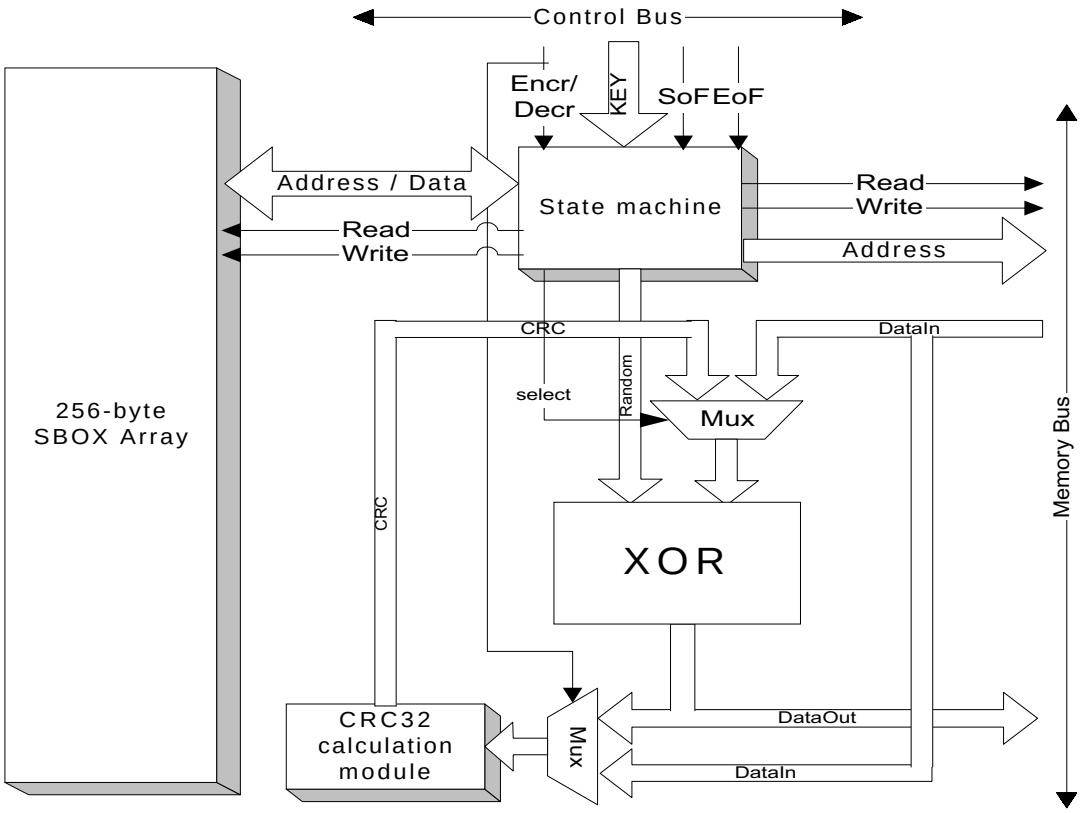

Fig. 3 *Block diagram of the WEP implementation*

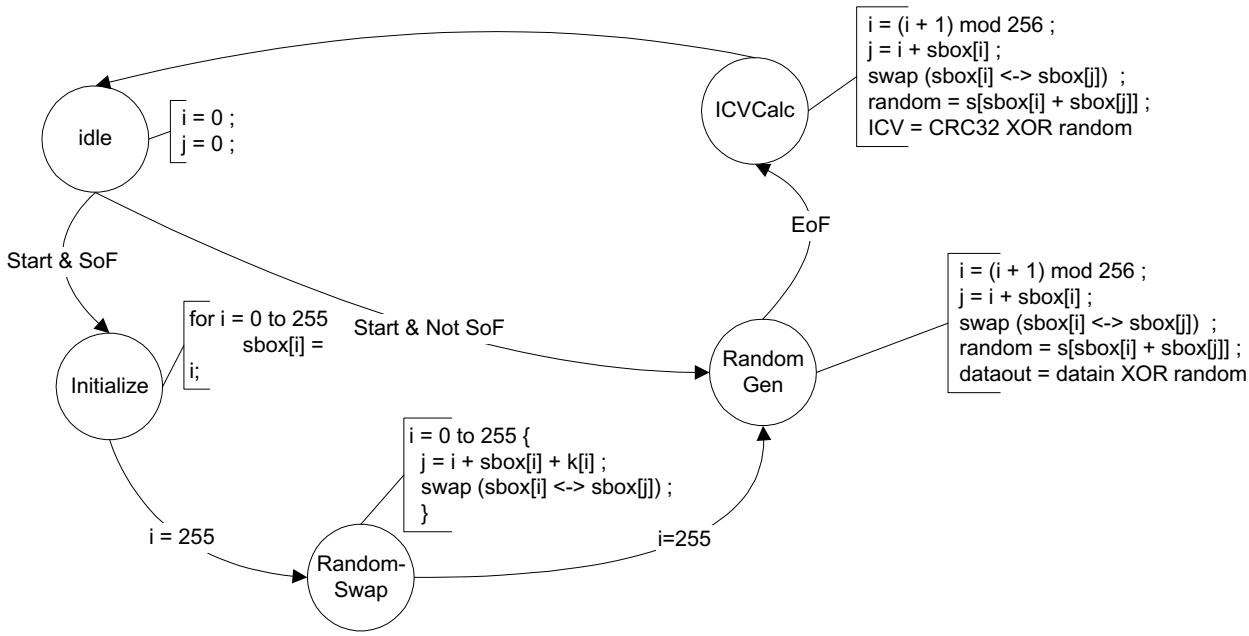

Fig. 4 *The WEP state machine*

In the ***idle*** state, the counters that index the SBOX array are initialized to 0.

In the ***Initalize*** and the ***RandomSwap*** states, the first two steps of RC4 algorithm are performed. The *Initalize* state needs 256 clock cycles to be completed, whereas the *RandomSwap* state needs 1024 steps (1 step to read S[i], one step to read S[j] = S[S[i]+K[i]+j], 1 step to write S[i] with the contents of S[j] and 1 step to write S[j] with the contents of S[i]). These two initialization steps are performed when a SoF is received otherwise the state machine proceeds to RandomGen state (and uses the previously initialized SBOX array).

In the ***RandomGen*** state the random numbers are generated in 5 steps, and are XORed with data coming from *datain*. The data are then stored in the same location in memory. The plaintext data are also passed to parallel CRC module to calculate the CRC.

If an EoF signal is received, the state machine proceeds to ***ICVCalc*** state and the CRC data contained in CRC module are also XORed with random RC4 numbers and stored in the ICV register.

The steps that are followed in order to transmit/receive encrypted/decrypted data are the following:

a) The host processor programs the start address and size of the block to be encrypted/decrypted
b) Then it programs the cipher key, and asserts a Start command, plus a Start of Frame (SoF) or an End of Frame (EoF) command, in order to proceed to the initialization steps, or ICV calculation respectively,
c) The system waits the WEP module to finish, by polling the start bit or receiving an interrupt, and
d) The data are transmitted, or passed to the upper layer.

4 WEP Integration

The WEP module was implemented and integrated in the VirtualNet 802.11 Wireless LAN architecture as a hardware accelerator module for an ARM-based MAC controller. The WEP module was interfaced to the ARM Synchronous Bus [3] as illustrated in Figure 5. The major blocks of this architecture are the following:

- The ARM processor which controls all the functional blocks and performs some of the MAC layer functions,
- The host interface (i/f) controller which offers an interface for accessing the local SRAM to an external host.
- The lower level MAC functions such as TSF timers and CRC-32 calculation circuit.
- The external memory controller which offers an interface, to external flash/SRAM, and
- The WEP module.

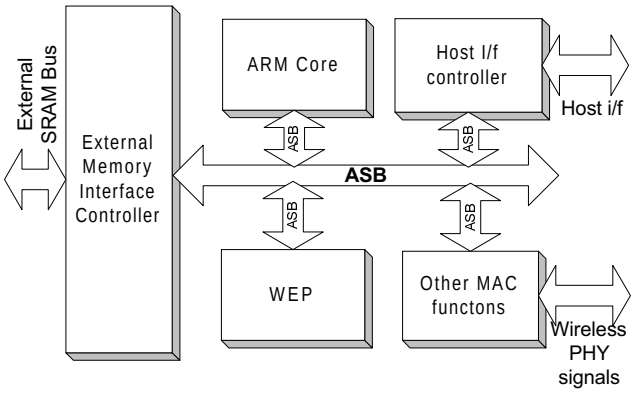

Fig. 5 *Virtual Net 802.11 architecture*

During a typical transmit process using the above architecture the following steps are performed:

a) the host i/f controller gains access of the ASB bus giving the opportunity to the host to fill the appropriate data buffers in SRAM,
b) when the host finishes the data preparation process, constructs the appropriate buffer descriptors and informs the ARM processor that a new transmission can take place,
c) the ARM processor reads the transmit buffer descriptors and schedules a transmission through the MAC functions block. If the requested transmission is an encrypted one, then ARM processor also programs the WEP module to encrypt the data block,
d) when the WEP module finishes, it appends the calculated Integrity Check Value and informs the ARM that the transmission can start, and
e) the MAC functions block transmits the encrypted data.

During the reception of an encrypted frame, the following process is performed:

a) The MAC functions block stores the received frame into a block of the external memory and informs the ARM processor that a complete frame has been received.
b) Then, the ARM processor checks the packet header in order to determine if this is an encrypted frame and in that case, it programs the WEP module with the appropriate cipher key,
c) The WEP decrypts the data in the external RAM and informs the ARM processor when the

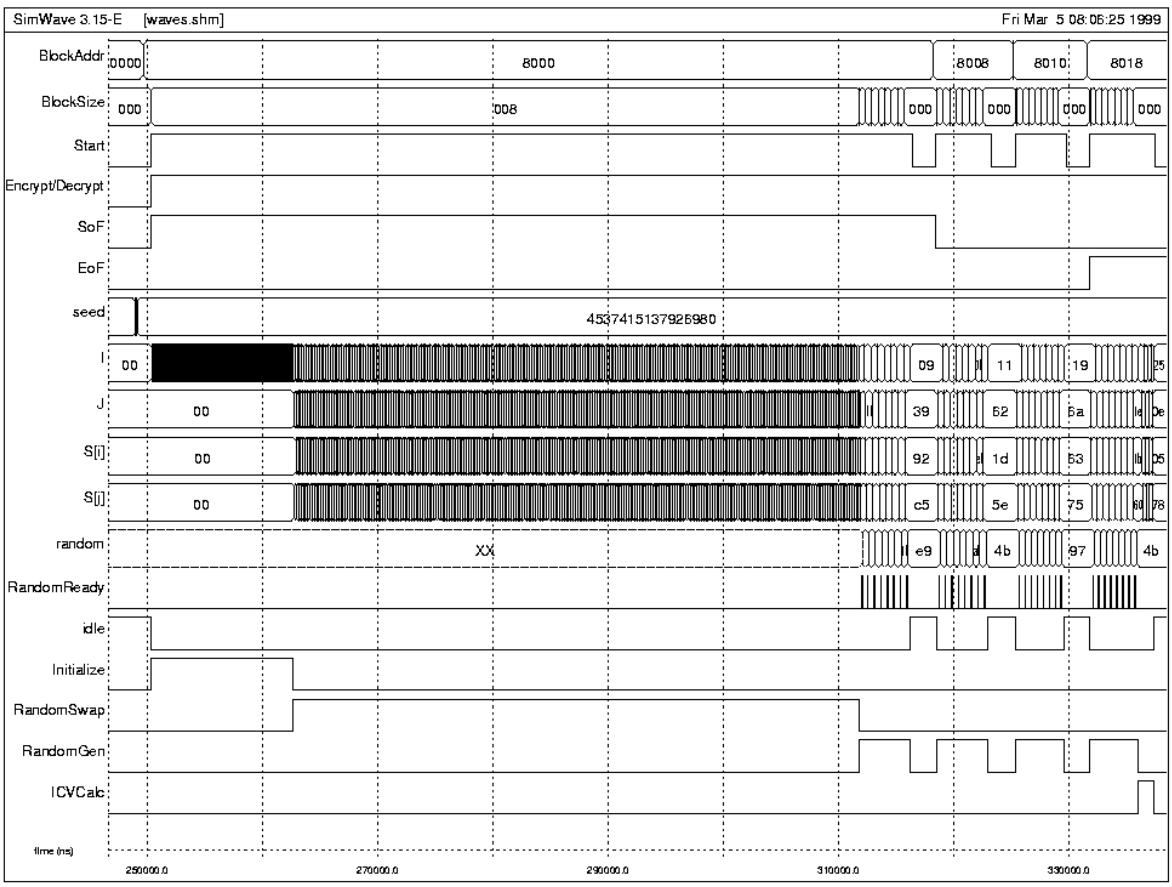

Fig. 6 *An encryption procedure example.*

decryption has been finished, and if the integrity check was passed,

d) Finally, the ARM processor constructs the appropriate receive buffer descriptors and informs the host to pump out the valid data.

A simulated run of the encryption procedure is illustrated in Figure 6. At time 250000 the start command forces the WEP machine to leave the *idle* state and get into the *Initialize* state. At time 263000 the state machine goes into the *RandomSwap* state until time 312000. Then the WEP module starts to produce the random numbers, (each RandomReady pulse indicates a valid random number). The WEP module encrypts the eight-byte blocks 8000-8007, 8008-8010, 8010-8018, 8018-8020, using successive start commands (without requiring re-initialization), and at the end, it proceeds to the *idle* state.

5 Conclusions

The presented hardware implementation of the WEP algorithm is more than 10 times faster than microcode implementations, while the silicon area cost was less then 4000 equivalent gates in ES2 0.5 μm technology. The WEP module was fully tested and behaves as required, as a hardware acceleration module in an IEEE802.11 WLAN MAC controller at 1 and 2 Mbps.

References:

[1] IEEE Std 802.11-1997: *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification"*.

[2] Bruce Schneier, *"Applied Cryptography, Protocols, Algorithms and Source Code in C"*, John Wiley and Sons, Inc., second edition, 1996.

[3] Advanced RISC Machines Ltd., *"Introduction to AMBA"*, Document No: ARM DVI 0010A, 1996