DESIGN OF A VARIABLE KEY LENGTH CRYPTOGRAPHIC PROCESSOR

*Bharathwaj S.V. *, Kishore L.N.**, Arulalan M R****
Department of Electronics and Communication Engineering
Sri Venkateswara College of Engineering, Sriperumbudur

ABSTRACT - Secrecy has always played a central role. Not just military applications, even day-to-day civilian applications like Internet require data to be transmitted through a secure network. Most software based cryptographic systems are not only easier to design & upgrade but also portable and flexible; but it has an inherent inefficacy - the level of security is often not sufficient for a number of applications such as e-commerce, e-mail and e-banking. Hardware implementations are by nature physically more secure as they cannot be easily read or modified by an eavesdropper. Hardware also has another advantage: in the encryption process the data is correlated according to an algorithm which usually performs operation on the same data, this characteristic prevents computer techniques such as out of order execution to improve performance of the software execution which is executed one instruction at a time; meanwhile hardware implementations tend to be extremely parallel in nature and therefore will run many orders of magnitudes better than a software implementation.

In this paper we propose a novel architecture for designing variable key length cryptographic processor based on Advanced Encryption Standard that will improve security. This novel architecture will provide a security that will increase as the cube of the present standard, which is most suited for the present industrial requirements.

Key Words: Cryptographic processor, Advanced Encryption Standards, Reconfigurability, Variable Key length.

1 THE ADVANCED ENCRYPTION STANDARDS:

The Advanced Encryption Standard (AES) specifies a NIST-approved cryptographic algorithm that can be used to protect electronic data .The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called cipher text; decrypting the cipher text converts the data back into its original form, called plaintext.

This standard specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. The AES involves a specific set of operations that are carried out a stipulated number of times as determined by one of the 3 data block or key lengths used. In this project we implement the Rijndael algorithm using a fixed data block length of 128 bits and utilize all the 3 different key lengths to encrypt the plain text.

The design of Rijndael are based on the considerations of the following [1]:

- Resistance against all known attacks
- Speed and code compactness on a wide range of platforms
- Simplicity of design

The round transformation is composed of three distinct invertible uniform transformations, called layers. The suggested layers have following functions:

- The linear mixing layer: guarantees high diffusion over multiple rounds.
- The non-linear layer: parallel application of S-boxes that have optimum worst-case non-linearity properties.
- The key addition layer: A simple EXOR of the Round Key to the intermediate State.

Before the first round, a key addition layer is applied. The motivation for this initial key addition is that any layer after the last key addition in the cipher (or before the first in the context of known-plaintext attacks) can be simply peeled off without knowledge of the key and therefore does not contribute to the security of the cipher. In order to make the cipher and its inverse more similar in structure, the linear mixing layer of the last round is different from the mixing layer in the other rounds.

Thus the security offered by AES against all the unauthorized attacks is dependable and that the flexibility in deciding the length of the keys to be used provides further a scope to enhance the security.

2 RECONFIGURABLE ARCHITECTURE:

Complex system designers are very much keen in providing a Reconfigurable architecture especially for the applications in Wireless Communication Systems. Reconfigurable hardware offers an acceptable trade-off between flexibility of programmable general-purpose processors and performance of a dedicated hardware designs. This characteristic qualifies reconfigurable hardware to be used in application areas showing a fast progression in the proliferation of standards, as it is the case for upcoming OFDM (Orthogonal Frequency Division Multiplexing) based WLAN standards like IEEE 802.11A/G.

The concept of a processor with dedicated hardware accelerators is afflicted with the low flexibility of the accelerating units. If standards supported by the hardware accelerator change, the chip may no longer provide the desired acceleration. A time consuming and therefore expensive redesign is required. The critical nature of the problem forces many manufacturers to use high performance general-purpose processors, which provide the desired flexibility regarding standard updates. The limitation is that due to their lack of specialization for the required tasks, they have poor component utilization and therefore require a large chip area that has a negative impact on the cost per chip.

Reconfigurable processors offer a possible solution for these problems. The processor allows an efficient realization of the control-flow dominated tasks, while the reconfigurable component can be used to speed up data-flow intensive tasks. In order to lower the inherent hardware overhead of reconfigurable solutions, the reconfiguration capabilities can be reduced to a special application area, resulting in a function-specific reconfigurable device [2].

3 SUITABILITY OF AES FOR RECONFIGURATION

In the present application that we propose, the AES employs a number of variations with regards to selecting the key length and the data block length. As for the latter is concerned, implementation of different block lengths is not justified from the economical point of view, as it would increase circuit area and cost without any considerable gain in security. Hence we consider exploiting the variations in key length to encrypt a block of data. This would provide a multi-fold increase in security due to the complete randomness of the selection of key lengths.

Moreover reconfiguration can also be brought about by employing different algorithms to encrypt the data in addition to AES. The efficiency in terms of speed and area utilization is then determined by the operations employed in various algorithms that are common.

For the present case we restrict to a single algorithm that can be reconfigured to encrypt data using alternate key lengths of either 128,192 or 256 bits, as specified by the AES.

4 PROPOSED ARCHITECTURE

The organization of the processor is given in fig 1. The processor is reconfigured to encrypt the incoming data block in one of the three AES standards (AES-128, AES-192, and AES-256) by a random number generator. The selection of the block length for the data is determined by the value produced by a random number generator. This is a Pseudo Noise (PN)-sequence generator of 3 bits width that will decide the length of the data block as well as the cipher key.

This will also decide upon the number of rounds the operations have to be carried out in the encryption process. A PN sequence generator can be realized using a feedback connection of a series of flip –flops.

The output of the Pseudo Random Number generator will select one of the entries stored in the Look-Up Table (LUT).

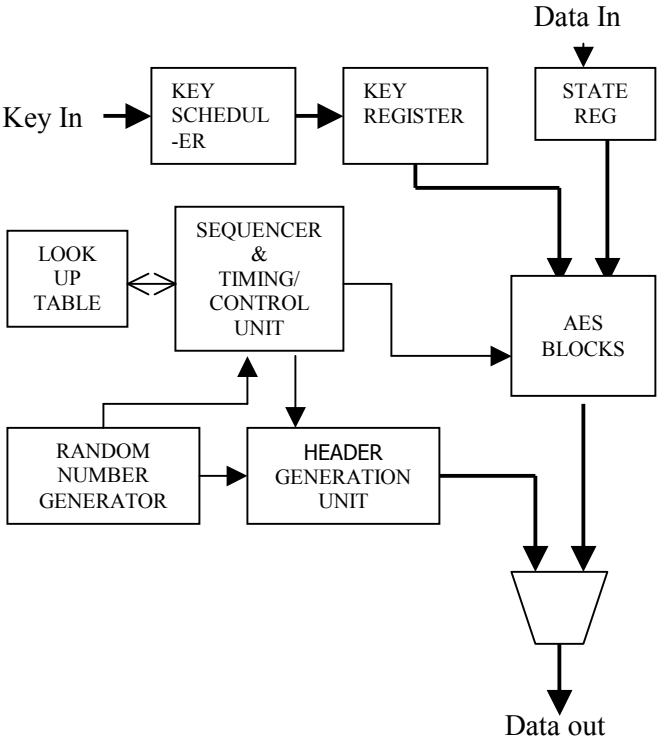

Fig 1 : Proposed Architecture

The sequence of the operations to be carried out in the algorithm is stored in the LUT in the form of control signals. The entries in the LUT provide the bit pattern corresponding to one of the sub-operations to be performed in the encryption process. As specified by the AES standard there are four transformations to be carried out on the State array repeatedly for a specific number of rounds. The signals are derived from the LUT to enable the appropriate block, delegated to perform a specific sub-operation.

The heart of the architecture is the SEQUENCER & TIMING CONTROL UNIT. It is designed to route the signals obtained from the LUT to the specified blocks corresponding to the bit pattern .The sequencer also ensures that the data pass through the stages in the exact order as specified by the standards. As the ADD ROUND KEY operation requires the cipher key from the KEY SCHEDULER, the control unit also ensures the coordination between the encryption blocks and the key generator. The beginning and the end of the encryption process is signaled by the CONTROL UNIT. This unit also ensures that every block in the encryption unit has successfully implemented the transformation and that the processor is reconfigured according to the specific key length.

The core of encryption is carried out in the AES BLOCK. This is composed of 4 independent units that perform one of the operations as mentioned in the Rijndael algorithm. The SEQUENCER activates one of these units and correspondingly the transformation of the state is carried out by that unit .The ADD ROUND KEY block alone depends on the cipher key generated by the KEY SCHEDULER. Except for this, the other blocks operate on the contents of the state register and generate the intermediate cipher text. The AES BLOCK contains MIX COLUMN, ADD KEY, BYTE SUB and SHIFT ROWS blocks. It is interesting to infer from the algorithm that except for the MIX COLUMN operation the other transformation doesn't require a much rigorous hardware implementation [5]. The ADD KEY, BYTE SUB and SHIFT ROWS blocks perform a simple X-OR operation, memory fetch and a simple reordering of bytes respectively. Hence the number of clock cycles involved is comparatively lesser than that taken by the MIX COLUMN unit. Hence this unit requires a more efficient implementation in hardware [6]. We use a novel method here to perform the MIX COLUMN operation.

To maintain the synchronization, a header is incorporated with the encrypted data to identify the length of the cipher key. A HEADER GENERATION unit based upon the reconfiguration mode generates a stream of bits carrying information about the length of the key used for encryption. At the end of the encryption process the cipher text is appended with this header and together sent out as the encrypted data. The header can be further encrypted so that the information carrying the length of the cipher key can be camouflaged within the header. The KEY GENERATION unit performs the key scheduling and key expansion operation from the initial set of Nk words where Nk =Key length/32.The steps involved in key generation are very similar to those used for the data encryption. This effectively reduces the complexity involved in the generation of the key for every round of transformation.

A key register holds the round key to be used by the ADD ROUND KEY operation. A STATE REGISTER is provided for the storage of the intermediate states of the encrypted data at the end of each round.

5 HEADER GENERATION UNIT:

We have used standard architecture for the Rijndael algorithm and implementation. But the change here is the inclusion of reconfigurability and an additional Header Generation Unit. The former is explained in the previous section. It is worth explaining the architecture of the header generation unit.

As mentioned earlier, the 40-bit header performs two functions. It helps the receiver to identify the key length used in the encryption and it also adds to the encryption. The technique used here is called 'N-Marker Counter'. The number of ones in the 40bit header is the factor that determines the key length used to encrypt. For instance, the following table gives the key lengths and the number of ones used to indicate them:

Key length	Number of ones
128	11-17
192	21-27
256	31-38

Table 1 : Relation Between Key Length and Number of Markers in the Header

Bits `$b_0$` and `$b_1$` from the Random Number Generator `A` and the 3 bits from the `Local Random Number Generator` selected the row and column of the LUTI. The value read from the LUTI is the number of ones, which the header has to contain. This is then mapped to a 21x40 RAM to produce the header. Further rotation and shifting makes the header more secure. Latches, decoders and ROM are

used to implement the above architecture the signals b_0 and b_1 are input to this unit from the Random number Generator Clock is also an input signal from the control unit. Q_0, Q_1, and Q_2 are the three internally generated signals (from the Local Random Number generator) 'Hdr' is the header generator. Shown below is the RTL Schematic of the unit.

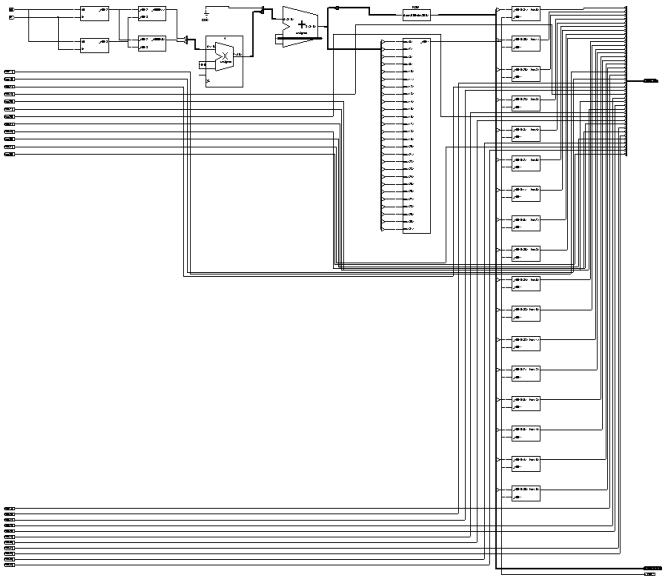

Fig 2: RTL Schematic of the Header generation Unit

6 PSEUDO RANDOM NUMBER GENERATOR:

As mentioned earlier, one of the three key lengths is chosen randomly. For this purpose we need a pseudorandom number generator. We use a type of random number generator called 'Linear Feedback Shift Registers'.

Shift register sequences are used in both cryptography and coding theory. There is a wealth of theory about them; stream ciphers based on shift registers have been the workhorse of military cryptography since the beginnings of electronics.

A feedback shift register is made up of two parts: a shift register and a feedback function. The shift Register is a sequence of bits. (The length of a shift register is figured in bits; if it is n bits long, it is called an n-bit shift register.) Each time a bit is needed all of the bits in the shift register are shifted 1 bit to the right. The new left-most bit is computed as a Function of the other bits in the register. The output of the shift Register is often the least significant bit. The period of a Shift register is the length of the output sequence before it starts repeating.

Cryptographers have liked stream ciphers made up of shift register: They are easily implemented in digital hardware. The simplest kind of feedback shift register is a linear feedback shift register, or LFSR. The feedback function simply performs the XOR of certain bits in the register; the list of these bits is called a tap sequence. Because of the simple feedback sequence, a large body of mathematical theory can be applied to analyzing LFSRs. LFSRs are the most common type of shift registers used in cryptography

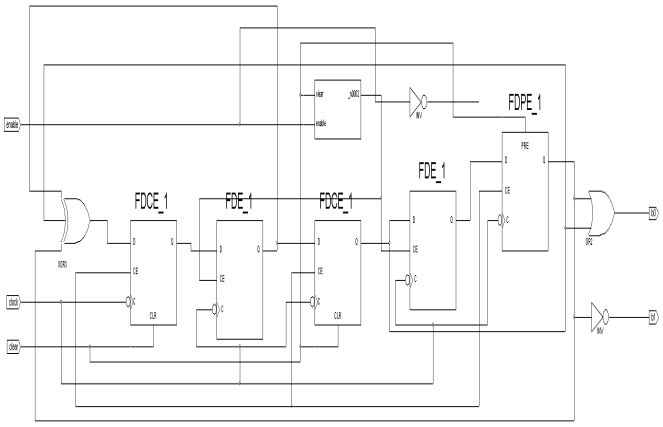

Fig 3: Pseudorandom Number generator

7 UNIQUE FEATURES OF THE CRYPTOGRAPHIC PROCESSOR:

The functioning of the 'CRYPTOGRAPHIC PROCESSOR' is based upon the control signals that are encoded in a Look-Up Table (LUT). The LUT is a ROM consisting of the information that enables the processor to determine which of the four sub-operations of the encryption process is slated for the next machine cycle. An LUT can effectively speed up the operations as the delay involved is that of simple memory access and with the LUT located on-chip this delay doesn't impede the time efficiency of the processor. Such an LUT based operation is also simple to integrate as the sequence of sub-operations are predefined and hence the control signals necessary to activate them can be accessed from LUT. This in contrast to the architectures based on Finite State Machine (FSM) [3] has its advantage of forestalling any unprecedented 'state conflicts', which are common in the FSM, based machines due to the asynchronous delay of the individual components that make up the FSM.

Implementing the AES in hardware has many advantages over software. The most significant achievement is the phenomenal increase in speed. In most cases hardware designed for a specific application can perform its task much faster than software running on a microprocessor.

A hardware implementation of AES can process 1.82 Gbits/sec using a 0.18 um CMOS library vs. 100 Mbits/sec for software running on a Pentium 200 Processor [4]. The speed of hardware is ideal for encrypting/decrypting Internet traffic at the physical layer that requires very high throughput so that bottlenecks don't occur. Although software provides greater flexibility, hardware provides greater efficiency. This efficiency extends to power dissipation. For applications such as mobile phones, hardware is a good choice because of the significant power saving which translates into extended battery life.

A hardware implementation also allows for physical separation of the encryption/decryption module from the rest of the system it is being used in. Physical separation provides greater security because data in the encryption/decryption process never gets seen or has the potential to be intercepted by any other processors in the system. Thus incorporating this processor in the physical layer of any network allows the total isolation of the data from the other top layers. Further any additions towards error checking and correction can be implemented in this layer. This has the potential of appreciably reducing the payload in the top layers thus once again facilitating the enhancement in speed as well as security.

Another striking feature of the proposed processor is the ability to reconfigure itself for different key lengths. Implementation using fixed length for both the cipher key and the data block may not utilize the flexibility inherent in AES to the maximum. As specified by the standard AES algorithm shall support at least one of the three key lengths 128, 192, or 256 bits (i.e., Nk = 4, 6, or 8 respectively). Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations. AES, by itself can guarantee security to a great degree and by reconfiguring it for different key lengths the security level achieved will be scaled sufficiently. Specifically, by using 3 different key lengths, the probability that the encrypted block of data will be successfully decoded into the plaintext will drastically get reduced to the cube of the probability of success for decrypting a single key length AES encrypted data block. Put in other words, the security offered by encrypting data block using 3 different key lengths is going to surge the security offered by single key length AES, S_f to its third power i.e. S_f^3. Thus, the provision offered by AES in specifying the lengths of the cipher key directly contributes to the enhancement of the security of the information to be transferred.

Additional security is imparted by the 'CRYPTOGRAPHIC PROCESSOR' through its HEADER GENERATION unit .The header which is chosen to be of 40 bit in length accomplishes the synchronization between the encryption and decryption process by incorporating the information of the key length used in the encryption process. In addition, it also performs additional encryption by not directly revealing the key length employed as the number and relative position of '1's which are the fundamental factors of determining the key length information are made completely random. In this way security is implanted in the entire block of data that originates from the 'CRYPTOGRAPHIC PROCESSOR'.

8 SIMULATION RESULTS:

Table 2: UTILIZATION OF SLICES, IOBs AND LUTs:

UNIT	**SLICES** AVLB-6912	**BONDED IOBs** AVLB-320	**INPUT LUTs** AVLB-13824
Byte substitution	189	332	136
Add round key	74	128	84
Mix columns	152	264	256
Pseudo random number generator	3	5	4

Table 3: UTILIZATION OF BUFFERS:

UNIT	**INPUT BUFFERS**	**OUTPUT BUFFERS**	**TOTAL**
Byte substitution	8	128	136
Shift rows	128	256	384
Add rnd key	256	128	384
Mix col	128	128	256
Rand num generator	2	2	4
Header Generation	0	40	40

9 PROPOSED ALTERNATIVES

The 'CRYPTOGRAPHIC PROCESSOR' can offer a great deal of security with its reconfigurability architecture. The employment of different keys of varying lengths to encrypt a block of data by itself can enhance the shielding of sensitive data against illegal attacks.

Keeping the underlying architecture intact, a number of different alternatives can bring about further enhancements. Firstly, the 'CRYPTOGRAPHIC PROCESSOR' presented in this project works on Rijndael Algorithm, which is an accepted as the AES algorithm by the NIST (National Institute of Standards and Technology). The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Moreover apart from the Rijndael Algorithm any of the algorithms approved by NIST can also be employed for the encryption and decryption of data. Before the Rijndael was recognized as the algorithm for the AES there were many candidate algorithms such as Two Fish, Blow Fish, Cast, Frog etc that finally lost to Rijndael [5]. So the CRYPTOGRAPHIC PROCESSOR can seek to implement all the above algorithms with the reconfiguration of algorithms done at random. This will enhance the security further but then the implementation cost is the prime factor in choosing the algorithms.

A closer look into the AES algorithm will clearly reveal that all its operations involve only simple steps such as memory read, rotation and modulo -2 addition (XOR operation). Thus the blocks that make up the encryption unit can be efficiently realized in hardware using any of the architectures suggested in various literatures. But, the MIX COLUMNS operation is one that needs attention as it can limit the speed of the encryption process. Once again the complication is simplified by making use of the mathematical background of the steps involved in this sub-operation. MIX COLUMNS is primarily based on Galois Field Arithmetic in which multiplication and addition are done in GF (2^8). Thus by incorporating an efficient GF multiplier the time consumption of the MIX COLUMNS operation can be greatly reduced. Some of the GF multipliers suggested in various literatures are Mastrovito multiplier, Massey-Omura multiplier, Hasaan-Bhargava multiplier etc. which try to cut down the time and area utilized in performing a GF multiplication. [6][7][8]

The KEY SCHEDULING operation also plays a major part in determining the time efficiency of the processor. A suggested alternative to the proposed encryption unit is one in which KEY SCHEDULING operation is carried out in parallel with the encryption process and utilizing the key for each round simultaneously as and when required by the encryption block. A 'pipeline' mechanism can also be implemented in which each operation of the encryption process can be overlapped. This will highly increase the throughput and the speed of the processor.

REFERENCES:

[1] J. Daemen and V. Rijmen, *AES Proposal: Rijndael*, AES Algorithm Submission, September 3, 1999

[2] R.W. Boderson N. Zhang. *Architectural Evaluation of Flexible Digital Signal Processing forWireless Receivers*. In Asilomar Conference on Signals, Systems and Computers, vol. I, oct 2000.

[3] Xinmiao Zhang and Keshab K.Parhi - I*mplementation Approaches for the Advanced Encryption Standard Algorithm*

[4] Kuo, Henry and Ingrid Verbauwhede - *Architectural Optimization for a 1.82Gbits/sec* VLSI implementation of the AES Rijndael Algorithm.

[5] M. A. Hasan and V. K. Bhargava. *Bit-serial systolic divider and multiplier for finite fields GF(2^m)*. IEEE Transactions on Computers, 41(8):972{980, Aug 1992.

[6] J. L. Massey and J. K. Omura. *Computational method and apparatus for finite field arithmetic*. U.S. Patent Application, 1981.

[7] M. A. Hasan and V. K. Bhargava. *Bit-serial systolic divider and multiplier for finite fields GF(2^m)*. IEEE Transactions on Computers, 41(8) :972{980, Aug 1992

[8] E. D. Mastrovito. VLSI Architectures for Computations in Galois Fields. PhD thesis, Link¨oping University, Dept. Electr. Eng., link oping, Sweden, 1991.