Hardware Implementation of Block-Cipher Scalable Encryption Algorithm

Shashank S Bhat¹
Software Design Engineer
Tektronix India Pvt. Ltd., Bengaluru.

Premananda B. S.²
Department of Telecommunication Engineering
R. V. College of Engineering, Bengaluru.

ABSTRACT

Secured communication of sensitive data has always been of prime importance in electronic devices. Information security, data integrity, confidentiality, authenticity and non-repudiation are the most important aspects of communication between electronic devices. The world today is moving towards embedded systems, increasing the need for information security with minimal resources. Scalable Encryption Algorithm (SEA) is one of the frontrunner to capture its share in embedded market. Scalable Encryption Algorithm is a parametric block cipher encryption technique used in resource constrained applications like sensor nodes and RFIDs as the main requirements of such applications are small code and memory size. In this paper, previous work on the Block-Cipher Scalable Encryption Algorithm and the advantages of SEA are compared to other light weight encryption algorithms are explained. The Scalable Encryption Algorithm design is tweaked to make use of 3-bit and 4-bit substitution tables and an eleven round, half-duplex, two substitution box SEA is implemented on Virtex-5 FPGA and verified using Xilinx ChipScope Pro. The design is also implemented using embedded c programming language and tested on a microcontroller test board to check for its working in the real world.

Keywords
Block Cipher, FPGA, Half-Duplex, Information Security, Scalable Encryption Algorithm, Substitution Table, Xilinx ChipScope Pro.
1. Introduction

In cryptography, encryption refers to a process of encoding messages so that only authorized users can access it. The world today is moving towards embedded technology and data transfer in these devices has to be secure. A robust algorithm is thus required to provide sufficient security under minimal availability of resources. SEA has proven to be the best suited algorithm for resource constrained applications. SEA is designed for low cost embedded environments having minimal resources with constraints such as memory size, processing power [1].

The design of SEA [1] aims at having less memory requirement and limited number of instructions. Thus, it uses basic bit operations like bit/word rotations, bitwise XOR, S-box and modular addition. In the design of SEA, the length of the key size is the same as that of the data block size and this block size is a multiple of 6b, where, b is the processor bus size. Hence for an 8-bit processor, a 48-bit or 96-bit key can be used. The scalability of SEA helps us to choose between a 48, 96, 144, ... -bit keys. Key size for an application is a tradeoff between security and requirement of resources. If a 96-bit key is used, the security provided enhances without affecting the performance since the operations are restricted to 8-bits. The key is derived on-the-fly. It consists of odd number of key rounds and the initial key is regenerated after specified odd number of rounds which reduces the memory required to store the key to maintain robustness.

Poschmann A. et al., [5] describes DES lightweight extension (DESL), an extended version of Data Encryption Standard (DES), which can be applied to devices with minimum resources such as RFID and smart cards. This uses same S-box repeated eight times to reduce the chip size to 38% less transistors as compared to the smallest DES implementation. D. Hong et. al., [6] provides the basics of High security and lightweight (HIGHT), a block cipher which works on a 64-bit data word and 128-bit key. It consists of two transformation phases (Initial transformation with 32 rounds using 4 sub-keys at a time and a final transformation) and a key schedule producing 128 sub-keys. The sub-keys are generated on the fly for encryption and decryption. Wheeler et al., [7] proposes Tiny Encryption Algorithm (TEA), a block cipher which works on 64-bit data and uses 128-bit key. It has 64 identical rounds. The main drawback in TEA is the duplication of keys (each key is equivalent to 3 others) which means the effective key size is only 126 bits.

The performance analysis of these algorithms can be found in Soren et al., [8] which compares the ciphers based on code size and throughput. The size of the cipher should be small as there is a lot of restriction on the size of SRAM and flash memory which are used for storing algorithm code and look-up tables. TEA is the smallest cipher while HIGHT uses the highest amount of program memory. A combined metric is devised which is a ratio of throughput and code size. DESL and HIGHT perform poorly in this metric, thus performing poorly in all metrics and hence do not provide comparable security. TEA suffers from equivalent keys which results in the reduction in security provided.

SEA [1-4] has been identified as a better option based on these lines. It is dependent on parameters like processor word size. It is advantageous as the encryptor and decryptor can be combined efficiently and also the keys are generated “on-the-fly”. It was developed for resource constrained applications like smart cards and sensor nodes. SEA code size is one of the smallest in
its category and in terms of throughput to code size ratio it outperforms DES and HIGHT. The scalability and robustness of the encryption key makes SEA the best alternate encryption technique for memory constrained applications. The use of SEA helps to overcome the limitations posed by other encryption techniques and establishes a strong encryption routine providing data security and safety from cryptanalytic attacks.

FPGA is chosen over microcontroller for implementation because it can provide a better performance per watt of power consumed. FPGA is suited where the problems are embarrassingly parallel and contain repetitive tasks. There are a large number of discrete I/O pins on FPGA which provides I/O flexibility which when combined with directly attached memories provides a better choice than a microcontroller. SEA is implemented on microcontrollers to check for its working in the embedded world where it finds real time applications.

We chose SEA as it looks promising for low power resource constraint embedded application and did slight modification in the design for better security. In section 2, the basic building blocks of SEA are explained. Section 3 talks about the design and implementation of SEA. Implementation of algorithm in FPGA as well as 8051 test board and its performance is discussed in section 4. Finally, conclusion and future scope is discussed in section 5.

2. Building Blocks of SEA

The building blocks of SEA refer to the individual blocks which when integrated into one work as the encryption/decryption algorithm. SEA\textsubscript{n,b} is based on a limited number of elementary operations which can be easily performed.

2.1 Parameters

SEA\textsubscript{n,b} operates on various text, key and word sizes. It is based on a Feistel structure with a variable number of rounds, and is defined with respect to the following parameters:

- \( n \): PlainText size, key size
- \( b \): processor (or word) size
- \( n_b = n/2b \): number of words per Feistel branch
- \( n_c \): number of block cipher rounds

2.2 Basic operations

The basic operations that are performed in the design of SEA are bitwise XOR, substitution box, word rotation, inverse word rotation, bit rotation and mod \( 2^b \) adder. The basic operations are mathematically defined as follows:

A. Bitwise XOR (\( \oplus \))

The Bitwise XOR is one of the simplest modules and as the name indicates it is used to perform XOR operations on two words, one bit at a time. The bitwise XOR is defined on \( n/2 \)-bit vectors as shown in equation (1).
B. Substitution Box S

The proposed design uses two types of S-box- 3-bit and 4-bit. The working of the two substitution box is discussed further.

3-bit Substitution Box

$SEA_{n,b}$ uses the 3-bit substitution box, with substitution values given by set $S_{T1}$. For efficiency purposes, it is applied bitwise to any set of three words of data. The 3-bit substitution box is designed using equation (2).

$$S_{T1} = \{0, 5, 6, 7, 4, 3, 1, 2\}$$

$$S: a \rightarrow a = S(a) \Leftrightarrow$$

$$a_{3i} = (a_{3i+2} \wedge a_{3i+1}) \oplus a_{3i}$$

$$a_{3i+1} = (a_{3i+2} \wedge x_{3i}) \oplus a_{3i+1}$$

$$a_{3i} = (a_{3i} \vee a_{3i+1}) \oplus a_{3i+2}, \quad 0 \leq i \leq \left(\frac{n}{3}\right) - 1$$

Where $\wedge$, $\vee$ and $\oplus$ respectively represent the bitwise AND, OR and XOR.

4-bit Substitution Box

$SEA_{n,b}$ uses the 4-bit substitution box, with substitution values given by set $S_{T2}$. For efficiency purposes, it is applied bitwise to any set of four words of data. The 4-bit substitution box is designed using equation (3).

$$S_{T2} = \{0, 9, A, B, 4, D, F, E, 8, 5, 2, 7, 1, 3, 6, C\}$$

$$S: a \rightarrow a = S(a) \Leftrightarrow$$

$$a_{4i} = (a_{4i+1} \wedge a_{4i+2}) \oplus a_{4i}$$

$$a_{4i+1} = (a_{4i+2} \wedge a_{4i+3}) \oplus a_{4i+1}$$

$$a_{4i+2} = (a_{4i} \wedge a_{4i+3}) \oplus a_{4i+2}$$

$$a_{4i+3} = (a_{4i+1} \vee a_{4i}) \oplus a_{4i+3} \quad 0 \leq i \leq \left(\frac{n}{8}\right) - 1$$

C. Word Rotation (R)

Word Rotation refers to the circular left shift of 8-bits of the word. It can also be referred to as byte rotation. The Word Rotation $R$ is defined on $n_b$-word vectors as in equation (4).

$$R: a \rightarrow b = R(a) \Leftrightarrow$$

$$b_{i+1} = a_i, 0 \leq i \leq n_b - 2$$
\[ b_0 = a_{nb-1} \]  

**D. Inverse Word Rotation (R⁻¹)**

Inverse Word Rotation refers to the circular right shift of a byte of the word. The Inverse Word Rotation R is defined on \( n_b \)-word vectors as in equation (5).

\[
R^{-1}: a \rightarrow b = R^{-1}(a) \leftrightarrow \\
\quad b_i = a_{i+1}, 0 \leq i \leq n_b - 2 \\
\quad b_{nb-1} = a_0
\]  

**E. Bit Rotation (r)**

Bit Rotation is a module where the word is divided into three parts, the higher, middle and lower byte. Cyclic right shift operation is performed on the lower byte of data while the middle part is retained. The left byte is subjected to cyclic left shift. On combining these three parts, the output is obtained. The bit rotation is defined on \( n_b \)-word vectors as in equation (6).

\[
r: a \rightarrow b = r(a) \leftrightarrow \\
\quad b_{3i} = a_{3i} \gg 1 \\
\quad b_{3i+1} = a_{3i+1} \\
\quad b_{3i+2} = a_{3i+2} \ll 1, 0 \leq i \leq \frac{n_b}{3} - 1
\]  

Where \( \gg \) and \( \ll \) represent the cyclic right and left shifts inside a word.

**F. Addition (Mod 2^b ⊕)**

Mod 2^b addition is a module where bitwise addition is performed. The mod 2^b addition (⊕) is defined on \( n_b \)-word vectors using equation (7).

\[
⊕: a, b \rightarrow c = a \oplus b \leftrightarrow \\
\quad c_i = a_i \oplus b_i, 0 \leq i < n_b - 1
\]  

### 2.3 Key Round

Key generation module is shown in Figure 1 and consists of individual modules like bit rotation, word rotation, substitution box and mod 2^b adder. The key used here is 48-bit in length with the seed key assumed. The key round is designed to generate the same key after designated odd number of rounds thus reducing the memory required to store the keys that are generated in each round.
Figure 1. Key Schedule and Encrypt/Decrypt SEA Round [1]

The key round is designed using equation (8)

\[ KL_{i+1}KR_{i+1} = F_K(KL_i, KR_i, C_i) \]

\[ KRI + 1 = KL_i \text{ XOR } R(r(S(KRI \oplus C_i))) \]

\[ KLI + 1 = KRI \quad (8) \]

2.4 Single Encrypt Round

The single encrypt round FE depicted in Figure 1 is designed using equation (9).

\[ L_i + 1, R_i + 1 = FE(L_i, R_i, K_i) \]

\[ R_i + 1 = R(L_i) \text{ XOR } r(S(R_i \oplus K_i)) \]

\[ L_i + 1 = R_i \quad (9) \]

2.5 Single Decrypt Round

The single decrypt round FD is depicted in Figure 1 and designed using equation (10).

\[ L_i + 1, R_i + 1 = FD(L_i, R_i, K_i) \]

\[ R_i + 1 = R^{-1}(L_i \text{ XOR } r(S(R_i \oplus K_i))) \]

\[ L_i + 1 = R_i \quad (10) \]
3. Design and Implementation

This section proposes a design of SEA with eleven rounds of encryption/decryption where both a 3-bit [3] and 4-bit [2] Substitution Box is used. The advantage of using two S-boxes is that it provides better security by selecting either the 3-bit or 4-bit S-box in each round depending on one of the bits of key. The proposed design is a half-duplex version of SEA with two substitution boxes.

3.1 SEA Design with Adaptive S-boxes

The proposed modification in the encrypt/decrypt block of SEA is achieved using the basic building blocks that are used in original implementation of SEA. The key generation module is same as that used in the normal design of SEA. In the proposed design, two substitution boxes: 3-bit and 4-bit are used as defined in equations (2) and (3). As a result of these modifications, the encryption structure changes with the key. Consequently, this setup increases diffusion of the key and hence it becomes very hard for cryptanalysis. In the proposed design, SEA 48,8 is implemented.

The 48 bit data is divided into two blocks each being 24-bits in length. The LSB block is added with right part of the key which is of 24-bit length in the first six rounds and the left part is added in the next 5 rounds. After this stage, the resultant is substituted using either 3 or 4-bit substitution box and the key decides which among the two S-Box has to be used in the operation. Bit rotation is then performed on the substituted block of data. The output of the bit rotation is XORed with word rotation output of the 24-bit left part of the PlainText in the encrypt operation while the inverse word rotation is used for the decryption operation. The proposed design is as shown in Figure 2.

3.2 Hardware Implementation of SEA

SEA was implemented on FPGA first by building individual components given by equations 1 to 7 using VHDL and then integrating all the blocks to create encryptor, decryptor and key round. For the implementation of SEA design on FPGA, the conventional method of dumping the VHDL/Verilog code onto the FPGA cannot be adopted due to bounded I/O constraints. Eventhough 188 I/O pins are available on Virtex-5 FPGA, they cannot be directly accessed to input data. Also, a 48-bit logic analyser is required to read the outputs. Thus, Chip Scope Pro, a Xilinx ISE tool is used for design validation where the outputs of the FPGA are tapped and displayed on the monitor. The FPGA used for validation of the design is Virtex-5: XC5VLX110T device. Chip Scope Pro Integrated Logic Analyser (ILA) core is software based customizable logic analyzer core that can be used to monitor any internal signal of a specific design. It includes many advanced features of modern logic analysers, including boolean trigger equations, trigger sequences and storage qualifications.
Figure 2. Proposed block diagram of SEA 48,8 Encrypt/ Decrypt with 3 and 4-bit S-boxes

Integrating the “integrated controller core” (ICON), an “integrated logic analyzer” (ILA) and a “virtual input/output” (VIO) into the design, all the can be monitored closely. Xilinx ChipScope provides users with convenient software based interface for controlling the ILA including the trigger settings which provides a feature to view the waveforms. The Basic connection between the FPGA and the ChipScope Pro Analyzer is given in the Figure 3.

3.3 Implementation of SEA on Embedded System

The SEA algorithm is implemented on 8051 flash board. The basic setup for the embedded implementation of SEA is shown in Figure 4. The PlainText and key each of 48-bit are given as inputs and the CipherText which is of 48-bit is obtained as the output from the encryptor. The CipherText and the same key as in the encryptor are taken as inputs and the PlainText is obtained as the output of the decryptor. The codes are dumped onto the board through a UART cable and inputs are hardcoded. LCDs are used to display the final output. The baud rate is set at 9600 bps. The frequency is set at 11.0592 MHz. Philips P89v551rd2 microcontroller is used which has 64 kb of code memory and data memory.
The communication happens between transmitter (TX_FLASH BOARD) and receiver (RX_FLASH BOARD) using UART through RS232 protocol. The transmitter has Key and PlainText as inputs and CipherText as output. The second 8051 microcontroller receives the encrypted data through UART and the decrypted data is obtained as the output. The 8x2 LCD screens are used at each end to display the inputs and outputs.

4. Results and Discussion

The proposed encryption and decryption modules of SEA\textsubscript{48,8} are coded in VHDL, simulated using Xilinx ISE, synthesized using Xilinx XST, implemented on Virtex-5: XC5VLX110T FPGA device and validated for various input combinations. The proposed SEA\textsubscript{48,8} is also implemented on 8051 test board using embedded C to check the working on a live system communicating securely using RS232 port. 48bit data is used for validation.
4.1 Results of FPGA Implementation

Encryption, decryption results and synthesis report obtained after implementation on FPGA is discussed as follows.

A. Eleven Round Encryption

A ‘1’ is selected to perform the encryption operation. Figure 6 illustrates the encryption operation.

For Eleven Round Encryption Operation: Inputs:
Encrypt/decrypt: 1
PlainText: 21223242526
Encryption Key: 123456789ABC
Output: CipherText: D203F737A1A9

![Figure 6. Encryption Output](image)

B. Eleven Round Decryption

A ‘0’ is inputted in the Encrypt/decrypt to perform the decryption operation. The decryption operation is illustrated in Figure 7.

For Eleven Round Decryption Operation:
Inputs:
Encrypt/decrypt: 0
Input (CipherText): D203F737A1A9
Encryption Key: 123456789ABC
Output:
PlainText: 21223242526
Figure 7. Decryption Output

C. Synthesis Report

The Half-Duplex model of SEA is coded in VHDL, simulated using Xilinx ISE, synthesized using Xilinx XST and is implemented on Virtex-5: XC5VLX110T FPGA device using Xilinx ChipScope. The details of the synthesis is as below:

Number of Slice LUTs used: 965
Number of LUT-FF pairs used: 965
Number of bonded IOBs: 144
Maximum combinational path delay: 42.324 ns
Maximum operational frequency, $F_{\text{max}} = 11.813$ MHz
Throughput, $T = F_{\text{max}} \times \text{Number of output bits per clock cycle} = 567$ Mbps

4.2 Results of 8051 Implementation

A. Eleven Round Encryption

For encryption, the inputs given are PlainText (48-bit) and key (48-bit) and output obtained is CipherText (48-bit).

Inputs:
PlainText: HIRVCE
Key: 123456
Output:
CipherText: " öΩ%

B. Eleven Round Decryption

For decryption, the inputs given are CipherText (48-bit) and key (48-bit) with the output obtained as PlainText (48-bit).

Inputs:
CipherText: " öΩ%  
Key: 123456
The encryption operation is not limited to 48-bits of data but can be applied to text files having data in range of tens of MBs. When such a scenario is encountered, the large data is split into 48-bit blocks and encryption operation is performed on these small bits of data. The six bytes of data are encrypted one after the other and the CipherText at the end of each encrypt round is simultaneously stored into another file. The decrypt operation is performed on similar lines and the output of the decryption operation (PlainText) is stored onto another file.

C. Brute-Force Time Calculation

Brute-force attack refers to the act of guessing the encryption key from the CipherText. It is implemented using nested ‘for’ loops where each loop is run 10 times and the time taken to execute the code is noted. The time taken to break the key using brute-force attack is calculated using equation (11). The basic data required for the calculation of brute-force attack is as follows:

- Number of possible combinations = \( N = 2^{48} \)
- Number of keys checked = \( K_c = 10^6 \)
- Time taken to check \( 10^6 \) keys = \( T_{MK} = 354 \) s
- Time taken to check total number of keys, \( T_C \):

\[
T_C = \frac{N}{K_c} \times T_{MK} = 3615 \text{ years}
\]  

Thus, the time taken to practically break the key is found to be 3615 years which is unfeasible. As a result, the SEA cannot be a victim of brute-force attack.

4.3 Comparison

The proposed 11 round half-duplex design is compared with the normal design [2] and Loop Implementation [1] with respect to number of CLK cycles, frequency, delay and the number of LUTs and Slices and is tabulated in Table I. The proposed design shows improvement in delay when compared to the loop architecture. Comparison also shows a significant reduce in the number of LUTs and Slices of the proposed design over the other SEA architectures.

<table>
<thead>
<tr>
<th></th>
<th>Proposed Design</th>
<th>Normal Design</th>
<th>Loop Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Cycles</td>
<td>1</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Max. Frequency (MHz)</td>
<td>11.813</td>
<td>12</td>
<td>72</td>
</tr>
<tr>
<td>Delay (ns)</td>
<td>85.3</td>
<td>80.1</td>
<td>152.7</td>
</tr>
<tr>
<td>LUTs</td>
<td>965</td>
<td>1476</td>
<td>300</td>
</tr>
<tr>
<td>Slices</td>
<td>368</td>
<td>638</td>
<td>167</td>
</tr>
</tbody>
</table>
5. Conclusions and Future Scope

Scalable Encryption Algorithm, SEA_{n,b} is an encryption algorithm which finds its application in small embedded systems enabling Internet of Things. The major parameters that define the algorithm and help in its design are plaintext size $n$, key size $n$ and processor or word size $b$. The architecture of SEA provides added flexibility for its implementation with various sets of parameters such as plaintext, key size and bus size provided the plaintext and key size must be same size. The SEA design is implemented on Virtex-5: XC5VLX110T using Xilinx ChipScope Pro. The maximum operating frequency is found to be 11.813 MHz. Throughput is is found to be 567 Mbps. In the loop architecture [1], the time taken is very high thereby reducing the throughput by 11 times. SEA design is subjected to brute-force cryptanalytic attack to check for its robustness. The time taken to guess the key using brute-force attack is found to be 3615 years which is practically impossible.

Future work in the domain would include the FPGA implementation of the loop architecture of SEA, evaluation of SEA on higher end processors (ARM) and verifying the robustness of SEA for other cryptanalytic attacks like the square attacks, slide attacks and interpolation attacks. Applications such as Wireless Sensor Nodes and RFIDs can be rigged with SEA to evaluate the degree of security it can provide in the real world.

References


