Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 7 (2012) pp. 102-110

A New Efficient Symmetric Encryption Algorithm Design and Implementation

Lo'ai Tawalbeh*, Hend Al-Hajsalem, Tasneem Abu-Qtaish and Ayat Khatatbeh

* Director of the Cryptographic Hardware and Information Security Lab (CHiS),
All authors are with: Computer Engineering Department, Jordan University of Science and Technology,
Irbid, 22110, Jordan
tawalbeh@just.edu.jo, hindsaid2004@yahoo.com, ektaish@yahoo.com and ayatkh@yahoo.com

***Abstract*:**
Symmetric-key encryption is used in many applications, including protection passwords and Internet-based transactions. Due to the need for algorithms to provide the required security to today's applications, the Advanced Encryption Standard (AES) competition was announced by the NIST. It had five finalist algorithms (MARS, RC6, Rijndael, Serpent, and Twofish), where then the call for the AES was the Rijndael. In this paper, we build "ALT", a symmetric encryption algorithm that have higher security than those finalist algorithms by combining their strengths and avoiding their weaknesses, in an efficient structure to provide optimum security and performance. The ALT algorithm, the complete hardware design, and its implementation (using VHDL) are proposed. Experimental results (area, delay, and throughput) were obtained by synthesizing the design for the target FPGA technology.

***Keywords*:** secure symmetric encryption, high performance algorithm, FPGA implementation, AES.

I. Introduction

Symmetric-key encryption algorithms are fundamental in cryptography. They are used in a large variety of applications, including protection of the secrecy of login passwords, ATM PINS, e-mail messages, video transmissions (such as pay-per-view movies), stored data files, and Internet-distributed digital content. They are also used to protect the integrity of banking and point-of sale transactions, and in many other applications.

The DES cipher (Data Encryption Standard) [1], was the current standard for symmetric (shared-key) cryptography. It was developed by IBM in the early 70's. Although DES has provided a secure encryption algorithm for the past 25 years, its block-length and key-length limitations needed to be advanced for the new century as the data is getting longer and higher security levels are required in today's applications [2].

In response to a growing desire to replace DES, the National Institute of Standards and Technology (NIST) announced the Advanced Encryption Standard (AES) program in 1997 to select a symmetric-key encryption algorithm where it announced a formal call to build symmetric secure algorithms under a number of certain conditions. In 1998, NIST announced the acceptance of fifteen candidate algorithms and requested the assistance of the cryptographic research community in analyzing the candidates. Among these fifteen algorithms, only five made it to the finalist (MARS, RC6, Rijndael, Serpent and Twofish) [3], where then the call for the AES algorithm was the Rijndael.

We investigated the five finalist algorithms deeply and found out the strength points and weaknesses of each of them. Then, all the main parts that helped improving the performance and security in these algorithms and made them be within the finalist algorithms are merged in one algorithm in an efficient structure. This paper shows in details the complete structure of the new proposed algorithm (we called it ALT - letters combinations from the authors names), and how it works and maintains it's security. Also, the hardware design that implements ALT, and its performance that is obtained by the experimental results are shown. ALT is designed to meet the requirements of the AES cipher, of being secure enough for the current and next ten years applications, good performance, simplicity, and reasonable cost.

ALT as any other symmetric-key encryption algorithm, has a key setup procedure which is a very crucial step in determining the security level of the algorithm. So, we paid attention to have a very efficient (and simple at the same time) key setup procedure. The ALT key setup is similar to the RC6 key setup procedure. It is being used for years in RC5 and is used again in RC6 and there is no serious or known attacks registered on it till now, and it has been approved that it is simple, secure, and its implementation is easy either in software or in hardware.

ALT used in its basic encryption step (core), a very strong core similar to the one proposed in Mars algorithm. This core consists of eight forward and eight backward rounds, the strength of this core comes from its structure and the use of a function called Efunction which performs multiple manipulations on different portions of the algorithm, and uses two sub keys in its architecture. Also the manipulation on each portion of the data is independent of the other which provides a good confusion and resistibility against differential attacks.

To add extra security to ALT, we embedded a linear transformation function (based on the one in Serpent algorithm) to its core. This function consists of simple and straight forward transformations; such as Xoring, addition, shifting, and rotation operations. In addition to its simplicity and good performance on modern processors, the linear transformation function is found to have bounds on the probabilities of linear and differential attacks. More over, and to provide good diffusion on the data with least cost, a Pseudo Hadamard Transformation (PHT) is used in ALT (based on the Twofish algorithm). PHT is a simple addition operation that has huge impact on diffusing the data.

On the other hand , ALT used a non-linear transformation on data, this non-linearity is achieved using S-box's .S-box represents a table of constant values that is used in a form which is seems to be random, and these values are used to provide more diffusion on data. In ALT, we used variable data to determine which S-box value to use in each state, hence increasing the ambiguity of the encrypted data.

This paper shows in details the complete hardware structure of ALT and how it works and maintains it's security. The ALT design is described using VHDL hardware description language. Then the VHDL model was simulated using Mentor Graphics tools (ModelSim) for functional correctness. Finally, experimental results (area, delay, and throughput) were obtained by synthesizing the design for the target FPGA technology and compared with other algorithms.

Finally, ALT is supposed to be simple, secure, and efficient symmetric-key encryption algorithm. As for any new proposed encryption algorithm, we tried to perform a lot of theoretical study and testing to prove its feasibility and resistibility to attacks. We hope that ALT will do the purpose it was built for and provides high security level and proves its effectiveness on the current machines and be flexible to go with the coming future developments, and we believe that this only can happen if ALT is adopted and put in use.

The rest of the paper is organized as follows: Section 2 briefly reviews the five finalist algorithms made it to the AES competition. Section 3 presents the design principles and choices of ALT. The proposed algorithm in details and its main building components are presented in Section 4. The ALT implementation experimental results details are shown in Section 5. Section 6 concludes this work.

II. Literature Review

This section provides brief description for the five candidates that NIST selected for further analysis. Those are: MARS, RC6, Rijndael, Serpent, and Twofish. No significant security vulnerabilities were found for these candidates during the analysis, and each of these algorithms constitutes potentially superior technology.

The first candidate was MARS [4]. It is a 128-bit block size symmetric-key algorithm with variable key size, ranging from 128 to over 400 bits. It was designed to meet and exceed the requirements for a standard for symmetric-key encryption in the next few decades. The main theme behind the design of MARS is to get the best security/performance tradeoff by utilizing the strongest techniques available today for designing block ciphers. As a result, MARS provides a very high level of security, combined with much better performance than other existing ciphers.

The second cipher algorithm was RC6 [7]. The design of RC6 began with a consideration of RC5 as a potential candidate for an AES submission. Modifications were then made to meet the AES requirements, to increase security, and to improve performance. While no practical attack on RC5 has been found, the studies provide some interesting theoretical attacks, generally based on the fact that the "rotation amounts" in RC5 do not depend on all of the bits in a register. RC6 was designed to thwart such attacks, and indeed to thwart all known attacks, providing a cipher that can offer the security required for the lifespan of the AES.

The third candidate was Rijndael [9] which was chosen to be the Advanced Encryption Standard (AES) by the U.S. government. It is expected to be used worldwide and analyzed extensively, as was the case with its predecessor, the Data Encryption Standard (DES). It became effective as a standard May 26, 2002. As of 2006, AES is one of the most popular algorithms used in symmetric key cryptography. The algorithm has 128-bit plain text and 128, 192 and 256 bits keys. Its key setup is fast, and its memory requirements are low which allows it to perform well in memory-constrained environments. The straightforward design and the conservative choice of operations should facilitate its further analysis, and the operations should be relatively easy to defend against certain attacks on physical implementations.

Serpent was the fourth candidate [10]. It has a block size of 128 bits and supports a key size of 128, 192 or 256 bits. The cipher is a 32-round substitution-permutation network operating on a block of four 32-bit words. Each round applies one of eight 4-bit to 4-bit S-boxes 32 times in parallel. Serpent was designed so that all operations can be executed in parallel (Serpent achieves its high performance by a design that makes very efficient use of parallelism), using 32 1-bit slices. This maximizes parallelism, but also allows use of the extensive cryptanalysis work performed on DES.

The last ciphering candidate was the Twofish [11]. A 128-bit block cipher with variable-length key up to 256 bits. The cipher is a 16-round Feistel network. The F function made up of four key-dependent 8-by-8-bit S-boxes, a fixed 4-by-4 maximum distance separable matrix over GF(2^8), a pseudo-Hadamard transform, bitwise rotations, and a carefully designed key scheduling. Twofish can be optimized for speed, key setup, memory, code size in software, or space in hardware.

III. Design Principles and Choices

The ALT algorithm were designed according to many principles to maximize the efficiency, throughput, and security.

An important aspect of ALT is that its components are designed to permit extensive analysis, and make the algorithm easy to use. In every step of the design, we refrained from using operations and structures which seemed "too hard to analyze". Since most computers today (with some exceptions) use word-size of 32 bits, all the operations in ALT

are applied to 32-bit words. At the current state of the technology, this choice provides a good tradeoff between the ability to run the algorithm on computers which are available today and the ability to take advantage of larger word-size in future architectures.

ALT is a Type-3 Feistel network, and has a block length of 128 bits and word-size of 32 bits, it follows that each block consists of four words. Among the various network-structures which are capable of handling four words in a block, it seems that a type-3 Feistel network [13] provides the best tradeoff between speed, strength and suitability for analysis. Also, the encryption and decryption operations in ALT are designed to be symmetric, and it is secure against chosen ciphertext attacks as against chosen plaintext attacks.

Many operations were embedded in ALT. These include Additions, subtractions and xors. Table look-up are also used to provided security as it was used before for DES and many other ciphers. ALT uses a single table of 512, 32-bit words, called the S-box [14]. In principle, a carefully chosen S-box can provide good resistance against linear and differential attacks, as well as good avalanche of data and key bits. But it should be mentioned that implementing the S-box lookups in software is relatively slow, and hence, this is another motivation to move for the hardware implementation.

The Data-dependent rotations in ALT are combined with arithmetic operations (such as addition) to provide very effective resistance against linear cryptanalysis. On the other hand, fixed rotations are also used to place the data bits in certain positions. Both data dependent rotations and fixed rotations has fast software and hardware implementations. The mixed structure used in ALT provide better resistance against new (yet undiscovered) cryptanalytical techniques. Namely, a cipher consisting of two radically different structures is more likely to be resilient to new attacks than a homogeneous cipher, since in order to take advantage of a weakness in one structure one has to propagate this weakness through the other structure.

IV. The Proposed ALT

ALT is a 128-bit block cipher with a 128-bit key length, it takes as input, and produces as output, four, 32-bit data words. The cipher itself is word oriented, in that all the internal operations are performed on 32-bit words. This word-based structure makes it is easier to perform the logical and arithmetic operations on hardware, such as xoring and shifting. The general structure of the algorithm is shown in Figure 1.

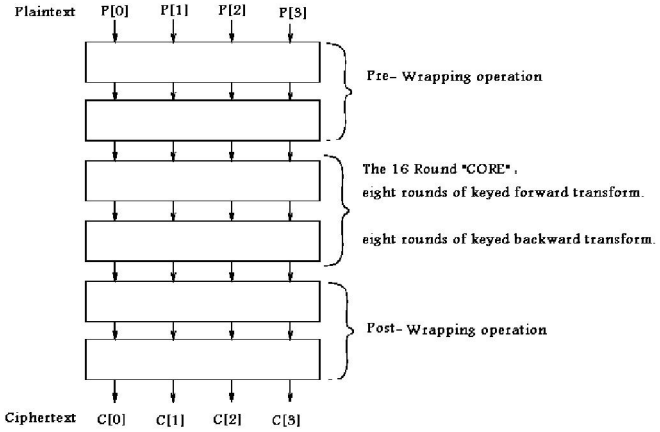


Figure1. The general structure of ALT

It can be realized from Figure 1 that the plaintext passes through a number of operations starting with the pre-wrapping which starts with a simple xoring operation (similar to pre-whitening that will be discussed later in this section). The wrapping process provides rapid mixing and key avalanche to harden chosen-plaintext attacks, and to make it harder to "strip out" rounds of the cryptographic core in linear and differential attacks [15][16]. After the wrapping, the input plaintext goes to the 16-round encryption process. The 16-round is called the MAR CORE, it consists of two parts: the superforward round and the backward round. These rounds are described in details through the documentation. After the CORE, the plaintext passes again through the wrapping which is quite similar to the one above but with slight differences.

ALT main components are key scheduling process, the pre-whitening and wrapping operations (including the Linear transformations and the Pseudo-Hadamard Transform (PHT)), and the main operations in the encryption core including the E-function. More details are provided below for each component.

A. Key scheduling

The ALT algorithm starts with running the key procedure because it is a "oneway" procedure, so it is difficult to infer supplied key from round keys. We start the schedule with the magic constants P32 = B7E15163 and Q32 = 9E3779B9 (hexadecimal). The value of P32 is derived from the binary expansion of e - 2, where e is the base of the natural logarithm function. The value of Q32 is derived from the binary expansion of Φ - 1, where Φ is the Golden Ratio. These values are used to initialize the values in the s-array which will lately be filled up with the 48, 32-subkeys. The key scheduling procedure is shown below:

Input: User-supplied b byte key preloaded into the c-word array L[0; : : : ; c - 1]
Number (r) of rounds
Output: w-bit round keys S[0; : : : ; 47]
S[0] = Pw
for i = 1 to 47 do
S[i] = S[i - 1] + Qw
A = B = i = j = 0
v = 3*max(c=4, r=48) = 144

for s = 1 to v do
{
A = S[i] = (S[i] + A + B)<<<3
B = L[j] = (L[j] + A + B)<<<(A + B)
i = (i + 1) mod r
j = (j + 1) mod c }
Algorithm 1. Key scheduling procedure

The dominant loop in modified RC6 key setup is the last for-statement loop in Algorithm 1. For b = 16 (number of input bytes) and r = 16 (number of rounds), the number of iterations in this loop is v = 3 * max(16 * 2 + 4; b=4) = 144, which is independent of b. So the estimates we make will be suitable for all key lengths of particular interest in the AES submission. Each iteration in the loop uses four 32-bit additions, one rotate to the left by three times, and one variable rotate to the left by r (if we consider r = (A+ B)) [7].

B. Pre-whitening and wrapping operation

Figure 2 shows the pre-whitening and wrapping operations. The whitening has a role in helping to secure the data; even it is simple, it helps increasing the difficulty of keysearch attack against the cipher text. The wrapping operation is part of the wrapper where it uses the "xor" operation. It xor's the first four key output words with the 128 bit plaintext (pair-wise xoring). Then, the data is swapped (shifted by one block to the left (32 bits)), and after that, they enter the LinearTransform block again.

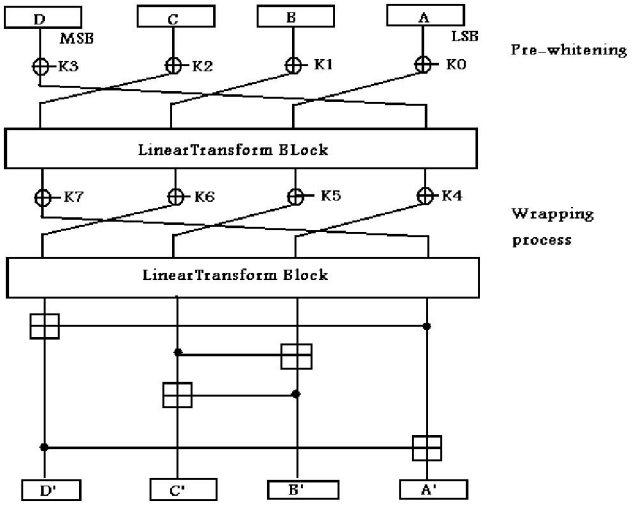


Figure 2. Pre-whitening and wrapping

The Linear Transform is simple and easy implement in a way that doesn't affect the performance. It contains operations such as xoring, shifting and rotations. These operations are considered simple and fast to operate in software and hardware. Even though they aren't meant to give cryptographic strength, they provide extra security to the algorithm by mixing the data together in a way that the operations through the cipher will not communicate and so, the data might not be exposed to the attacker.

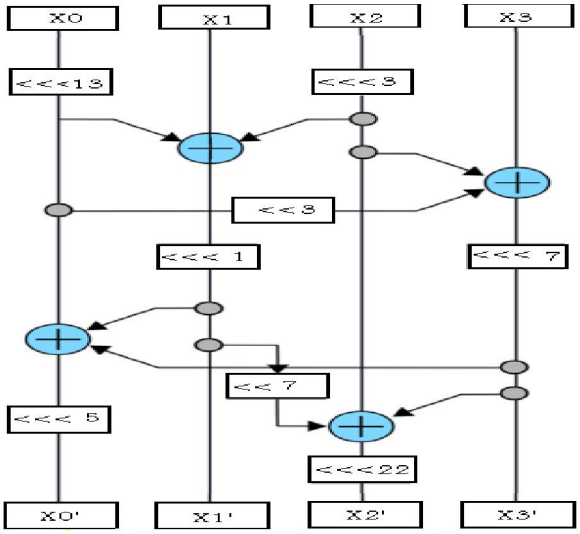


Figure 3. Linear Transform

These operations can be described by the following equations (where all thedata blocks –Xs are 32-bits):
X0,X1,X2,X3: (X0: least significant data block; X3: most significant data block)
X0 := X0 <<< 13
X2 := X2 <<< 3
X1 := X1 ⊕X0 ⊕X2
X3 := X3 ⊕X2 ⊕ (X0 << 3)
X1 := X1 <<< 1
X3 := X3 <<< 7
X0 := X0 ⊕X1 ⊕ X3
X2 := X2 ⊕ X3 ⊕ (X1 << 7)
X0 := X0 <<< 5
X2 := X2 <<< 22 (where <<< is Rotation ; << is shifting)

After going through the LinearTransform, the output words are xored with the next four subkey's K4, K5, K6, and K7 (wrapping process), and then they data swapped (shifted by one block to the left) and then enter the LinearTransform again as clearly can be seen from Figure 2. After that and before the data enters the encryption core, it is applied to the Pseudo-Hadamard Transform (PHT) operations. The PHT operation is considered simple and quick. It has been used in the TwoFish algorithm where the idea is originally was taken from the BlowFish algorithm.

The PHT operation involves addition of two 32 bit inputs (A and B) to get A' and B' as follows:
A' = A + B mod 232
B' = A + 2B mod 232,
In ALT, the PHT is applied on four 32 bit blocks (A, B, C, D) where A and D are added together, and B is added with C in the same way as shown in Figure 4. The resulting equations are as follows:
A' = A + D mod 232 C' = C + B mod 232
D' = A + 2D mod 232 B' = C + 2B mod 232

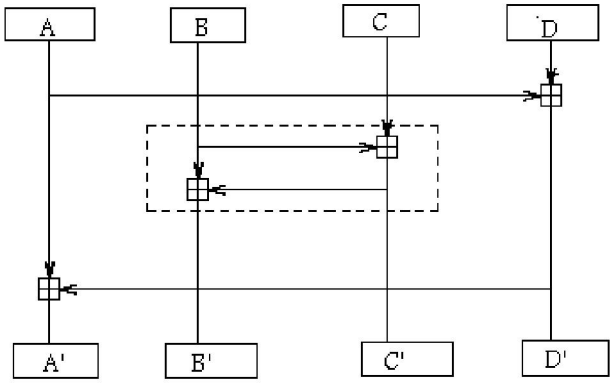


Figure 4. PHT operation used in ALT

After that, the resulting four words (A', B', C', D') are sent to the encryption core (CORE), to start the encryption process.

C. *Encryption Core*

The ALT encryption core is built on top of the MARS core. It was modified and optimized for better performance. The CORE is separated into two parts: the Superforward Round and Backwardsuper Round. The reason it contains two parts is to make sure that the encryption and decryption have the same strength, so the first eight rounds are performed in "forward mode" while the last eight rounds are performed in "backwards mode". The ALT core (CORE) is a type-3 Feistel network, and consists of sixteen rounds. In each round we use a keyed expansion function (E-function) which is based on a combination of xoring, addition, data-dependent rotations, and an S-box lookup. This function takes as input one data word (32-bits) and returns three data words as outputs.

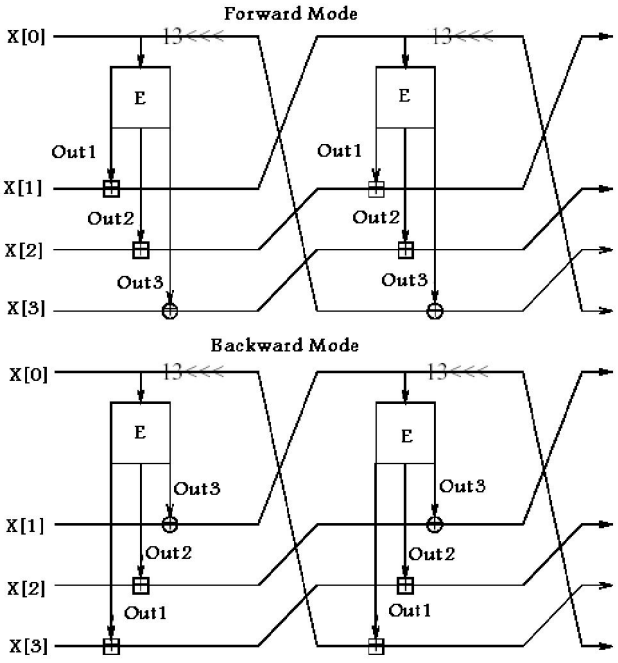


Figure 5. The Encryption CORE

D. *The E-Function*

The E-function used in ALT is shown in Figure 6.

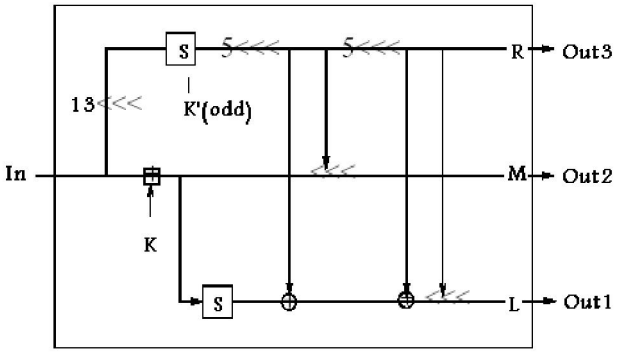


Figure 6. The *E-Function* of the 16 round CORE.

The E-Function takes as input one data word and uses two more key words to produce three output words. In this function we use three temporary variables, denoted below by L, M and R (for Left, Middle and Right). Below, we also refer to these variables as the three "lines" in the function. Initially, we set R to hold the value of the source word rotated by 13 positions to the left, and we set M to hold the sum of the source word and the first key word. We then use the lowest nine bits of L as an index to a 512-entry S-box and set L to hold the value of the corresponding S-box entry. We then xor the source entered to R with the second key (constrained to contain an odd integer) and then we view the lowest nine bits of the output from the xoring and place it as an index to the same 512-entry S-box, and place the output to R. After that, R is rotated by 5 positions to the left (so the 5 highest bits of the product becomes the 5 lowest bits of R after the rotation). Then we xor R with L and use the five lowest bits of R as a rotation value between 0 and 31, and rotate M to the left by this value. Next, we rotate R by 5 more positions to the left and xor it into L. Finally, we again use the five lowest bits of R as a rotation amount and rotate L to the left by this amount. The first output word of the E-function is L the second is M and the third is R.

E. *Wwrapping operation and post whitening*

The wrapping operation here is considered as the same one discussed in the beginning of this section, with some modifications The wrapping does not contain the PHT operation so the last output here will be from the second LinearTransform. And the keys xoring will use the last eight sub keys [K40 – K47], so the post whitening is considered within the operation and the cipher text should be cleared output as we can see through Figure 7.

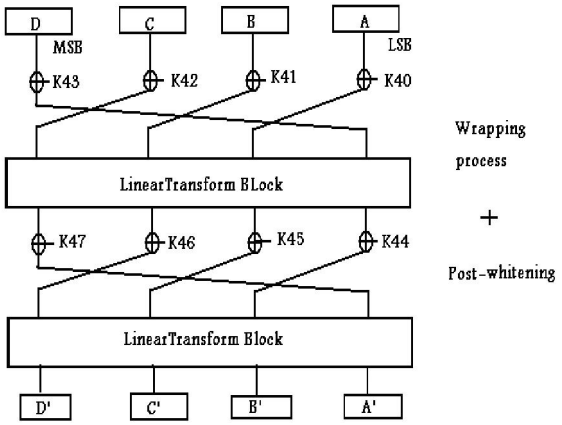


Figure7. Wrapping and post-whitening

F. Decryption process

As we mentioned before, the ALT decryption process is almost the "inverse operation" of the encryption process. This means that the first wrapping is almost the same as the second wrapping to provide security against chosen ciphertext attacks as against chosen plaintext attacks. The decryption of the pre-wrapping is quite simple, where the LinearTransform is used in the reverse order as the subkeys will be reversed. The PHT decryption can be easily obtained by reversing the encryption operations. Appendix A provides the Psudo-code for encryption and decryption operations.

The decryption of the CORE is similar (not identical) to the encryption. We provide a pseudo-code for decryption and encryption in Appendix B. Finally the decryption to the post-wrapping is also as mention in the pre-wrapping; the LinearTransform is used in reverse order as the subkeys are used in reverse order too.

V. ALT Security

ALT algorithm was built based on the research we did by deep studying and analyzing the finalist algorithms for the AES competition, taking in consideration their security and performance.

The design and selection of ALT operations were done carefully to maximize the security and performance. Starting with the wrapping, which helps assuring the ALT security by having different role than the middle 16-Rounds. The wrapping operations aren't supposed to provide complete security against any attack, but they help in diffusing the data entering the structure and hardens certain cryptanalytical attacks. The objective of the wrapper is to make as most avalanche as possible to the data, and this is why we choose the LinearTransform. Along with the LinearTransform, the PHT block also affords diffusion to the data and that makes the wrapper a highly diffusion provider block in the ALT structure.

The security in the 16-Rounds encryption CORE should be high against known attacks. The security of the core was proved within a number of studies from the IBM Corporation [6].The number of rounds has its role in providing high security level to the structure. Studies suggested 11 rounds to highly secure the core against known attacks, so using 16 rounds in the ALT is perfect choice that takes in consideration the ecurity/performance tradeoff. Within the rounds, the E-Function has its own advantage. It was built in a way where the output data are almost independent from each other and what backs them up is the use of the S-box, which in turns has its own advantage in security; because of its resistance to different attacks. And the S-boxes were among the main reasons for the high security in DES.

Another feature to make the algorithm more secure is by using rotations that depend on data just as we used in the E-function and key schedule process. With a sufficient number of rounds, it could provide great confusion and diffusion. The key schedule was chosen because of its high security that can be summarized in the following points:

- Key expansion is identical to that of RC5 and RC6 and no known weaknesses.
- No known weak keys.
- No known related-key attacks.
-Round keys appear to be a "random" function of the supplied key.
- Bonus: key expansion is quite "oneway" which is difficult to infer supplied key from the round sub keys.
-The best attack appears to be exhaustive search for the user-supplied encryption key [9].

VI. Experimental Results

The ALT algorithm was described in a hierarchal bottom-up structure and based on separate modules. This module-based design makes it easy for implementation, tracing and debugging, and scalability.
The ALT design is described in VHDL (VHSIC Hardware Description Language), and then simulated using Mentor Graphics tools (ModelSim) for functional correctness. The design was synthesized using Xilinx ISE 9.1i for Spartan3A FPGA chip (target device is xc3s1400an) to obtain delay and area results.

This section shows the simulation and synthesis results

A. Simulation Results

In this section, we provide snapshots of the simulation results of ALT. They show the inputs, outputs, and intermediate signals.

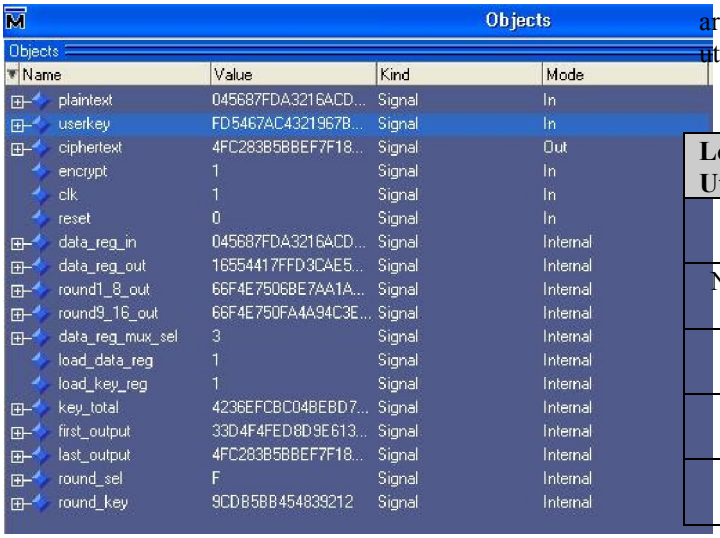


Figure10. The design signals

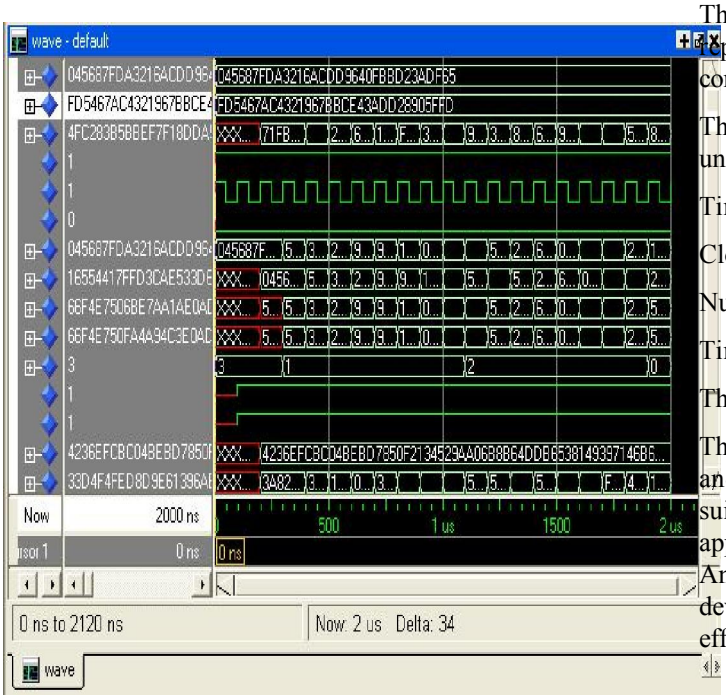


Figure11. The simulation process

B. *Synthesis Results*

The ALT design was synthesized using Xilinx ISE 9.1i for Spartan3A FPGA chip (target device is xc3s1400an) to obtain delay and area results.

The critical path delay of the ALT design was obtained to be 22.63 ns, which means that the maximum operating frequency of the design is 44.18 MHz. On the other hands, the area was found to be 31,024 slices. Finally, the device utilization is summarized in Table1

Logic Utilization	Used	Availabl e	Utilizatio n
Number of slices:	3102 4	11264	275%
Number of slice Flip-Flop:	1309	22528	5%
Number of 4 input LUT's:	5569 1	22528	247%
Number of bonded IOBs	325	502	64%
Number of GCLK:	1	24	4%

Table1. Device utilization summary

The throughput of ALT (a measure of performance that represents the amount of data processed in the time unit) is computed as follows:

Throughput = Data unit (in bits) / Time (bps), where data unit = 128 bits,

Time = number of clock cycles * clock cycle time

Clock cycle time = 22.634 (minimum period) ns

Number of clock cycles = 20 cycles (from simulations), then

Time = 22.634 * 20 = 452.68 ns, which yields to:

Throughput = 128 / 452.68 = 0.28276 Giga bps

The above results give reasonable tradeoff between security and delay and area. In other words, the proposed ALT is suitable for providing high level of security for dedicated applications where area is important with acceptable speed. And so, ALT can be embedded in special purpose hardware devices to provide the required level of security with high efficiency and throughput.

VII. Conclusion

In this paper, we proposed a new symmetric encryption algorithm that have the same or even higher security and quality than the five finalist algorithms to the AES competition. The new algorithm is called ALT based on letters combinations taken from the authors names. The ALT algorithm combines the strengths of the five finalists and avoids their weaknesses. In other words, all the main parts that helped improving the performance and security in these algorithms and made them be within the finalist algorithms are merged in one algorithm in an efficient structure.

This paper showed in details the complete hardware structure of the ALT and how it works and maintains it's security. The ALT hardware design was described using VHDL hardware description language. Then the VHDL model was tested using MentorGraphics tools (ModelSim) for functional correctness. Finally, experimental results (area, delay, and throughput) were obtained by synthesizing the design for the target FPGA

technology and compared with other algorithms. The results showed that ALT provides high level of security at high throughput without increase in area and delay. This makes the proposed ALT is suitable for providing high level of security for dedicated applications taking in consideration a reasonable tradeoff between the area and speed. And so, ALT can be embedded in special purpose hardware devices to provide the required level of security with high efficiency and throughput.

Acknowledgment

The authors would like to thank the CHiS in Jordan University of Scicen and Technology, and the Scientific Research Support Fund at the Ministry of High Education in Jordan for supporting this research.

References

[1] Data Encryption Standard. FIPS-PUB 46-3. USA, 1999.

[2] L. A. Tawalbeh and Q. Abu Al-Haija. "Enhanced FPGA Implementations for Doubling Oriented and Jacobi-Quartics Elliptic Curves Cryptography". Journal of Information Assurance and Security, Vol.6, Issue 3, pp. 167-175, Dynamic Publishers, Inc., USA, June 2011.

[3] Status Report on The First Round of The Development of The Advanced Encryption Standard. Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology. USA 1999.

[4] C. Burwick and eta. al., MARS - a candidate cipher for AES. IBM Corporation- USA,1999.

[5] S. Langford and M. Hellman, Differential-linear cryptanalysis. In Advances in Cryptology | Crypto '94, pages 17-25, Springer, 1994.

[6] Comments on MAR's linear analysis. The IBM MARS team. USA, 2000.

[7] R. Rivest, M. Robshaw, R. Sidney, Y. Yin. The RC6 Block Cipher. RSA Labs. USA, August,1998.

[8] B. Kaliski Jr and Y. Yin, On the Security of the RC5 Encryption Algorithm. RSA Laboratories Technical Report, TR-602 Version 1.0. September 1998.

[9] Rijindael, Advanced Encryption Standard. FIPS-PUB 197. USA, 2001.

[10] R. Anderson, E. Biham, L. Knudsen, Serpent: A Proposal for the Advanced Encryption Standard-Cambridge Univ., England, Univ. of Bergen, Norway. 1998.

[11] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, Twofish: A 128-Bit Block Cipher, Counterpane Labs, MN, USA. June1998.

[12] B. Schneier, Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Fast Software Encryption, Cambridge Security Workshop Proceedings. Springer-Verlag, 1994, pp. 191-204.

[13] H. Heys. Information leakage of Feistel ciphers. Information Theory, IEEE Transactions on. Volume: 47, Issue 1, pages: 23-35, Jan 2001.

[14] M. H. Dawson, Stafford E. Tavares, An Expanded Set of {S}-Box Design Criteria Based on Information Theory and its Relation to Differential-Like Attacks. Lecture Notes on Computer Science. Springer-Verlag. CI 547: page 352-367. 1991

[15] Q. Al-Haija and L. A. Tawalbeh. "Efficient Algorithms & Architectures for Elliptic Curve Crypto-Processor Over GF (P) Using New Projective Coordinates Systems". Journal of Information Assurance and Security, Vol. 6, Issue 1, pp. 63-72. Dynamic Publishers, Inc., USA, 2011.

[16] Y. Jararweh, L. A. Tawalbeh, H. Tawalbeh, and A. Moh'd. "Hardware Performance Evaluation of SHA-3 Candidate Algorithms". Journal of Information Security. Vol 3 No.2, pp 69-76, Scientific Research Publisher, USA. April 2012.

Appendix B. The CORE Encryption and Decryption Pseudo-code.

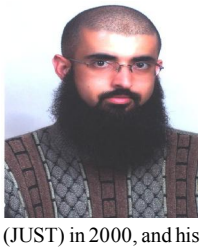
The Encryption of the 16- Round:

```
// Do 16 rounds of keyed transformation
 for i = 0 to 15 do
(out1; out2; out3) = E-function(D[0];K[2i + 4];K[2i + 5])
 D[0] = D[0] _< 13
 D[2] = D[2] + out2
 if i < 8 then // first 8 rounds in forward mode
 D[1] = D[1] + out1
 D[3] = D[3] _ out3
 else // last 8 rounds in backwards mode
 D[3] = D[3] + out1
 D[1] = D[1] _ out3 , end-if
// rotate D[ ] by one word to the right for next round
 (D[3];D[2];D[1];D[0])  (D[0];D[3];D[2];D[1]) end-for
```

The Decryption of the 16-Round:

```
// Do 16 rounds of keyed transformation
 for i = 15 down to 0 do
 // rotate D[ ] by one word to the left for this round
(D[3];D[2];D[1];D[0])  (D[2];D[1];D[0];D[3])
 D[0] = D[0] _> 13
 (out1; out2; out3) = E-function(D[0];K[2i + 4];K[2i + 5])
 D[2] = D[2] _ out2
  if i < 8 then // last 8 rounds in forward mode
 D[1] = D[1] _ out1
 D[3] = D[3] _ out3
 else // first 8 rounds in backwards mode
 D[3] = D[3] _ out1
 D[1] = D[1] _ out3 end-if  end-for
```

Author Biographies



Lo'ai A. Tawalbeh

Dr. Tawalbeh is the Director of the Cryptographic Hardware and Information Security (CHiS) lab at Jordan University of Science and Technology (JUST). He is a full time assistant professor at the computer engineering department at JUST, Jordan, and part time professor at NewYork Institute of Technology (NYIT)-Amman's campus, since 2005.

He got his BSc in electrical and computer eng. form (JUST) in 2000, and his Masters and PhD in computer engineering from Oregon State University (OSU) , USA in 2002 and 2004 respectively, under the supervision of prof. Dr. Cetin K. Koc , with GPA 4.0/4.0.

Dr. Tawalbeh has many research publications in many refereed international Journals and conferences. His research interests includes: information security,

hardware for cryptography and dedicated arithmetic algorithms for cryptographic applications, intrusion detection and computer forensics. Dr. Tawalbeh is a reviewer and a member of the editorial boards of many international journals and conferences in the area of the information security.

For more details, please see his website: www.just.edu.jo/~tawalbeh
Email: tawalbeh@just.edu.jo

Hind Al-hajsalem
is a researcher in computer engineering in the area of cryptography and information security. She holds masters degree in computer engineering.

Tasneem F. Abu-Qutaish.
Born in Doha/Qatar 15 Dec. 1984.
Attended Primary School at SunnyBank Primary School in United Kingdom till 1993. High School Education at Yarmouk University School in Jordan till 2002. B.Sc. Degree in Computer Engineering (specialization in Networking) at Jordan university of Science & Technology) graduated June 2007.
Two years experience in telecommunication path. Programming skills in C#, Java, Oracle 10g (familiar in JDBC-oracle database programming) and VHDL.

Ayat Khatatbeh. B.Sc. Degree in Computer Engineering (specialization in Networking) at Jordan university of Science & Technology) graduated June 2007.
Programming skills in C#, Java, Oracle 10g (familiar in JDBC-oracle database programming) and VHDL.