



Research Journal of  
**Information  
Technology**

ISSN 1815-7432



Academic  
Journals Inc.

[www.academicjournals.com](http://www.academicjournals.com)

## Compiler Optimization and Plain Text Pre-Processing to Hoist the Height of HIGHT in AVR Platform

Siva Janakiraman, K. Thenmozhi, John Bosco Balaguru Rayappan and Rengarajan Amirtharajan

Faculty, School of Electrical and Electronics Engineering, SASTRA University, Thanjavur, 613 401, India

*Corresponding Author: Siva Janakiraman, Faculty, School of Electrical and Electronics Engineering, SASTRA University, Thanjavur, 613 401, India*

### ABSTRACT

Microcontrollers are the small scale embedded devices that serves in most WSN in processing and communicating the data acquired by sensors. The AVR family encloses classy devices under 8 bit microcontroller category. Light weight crypto algorithms are designed to incorporate rational security in embedded devices. This study aspires in improvising the performance of 64 bit light weight block cipher HIGHT through software optimizations with AVRGCC compiler of Atmel Studio open source tool under windows platform. A modification on HIGHT algorithm is also proposed in this study to enhance its security level by additional manipulation on plain text prior to processing by the original algorithm. The coding is tailored towards decline in cost (memory requirement) and lift in performance (execution cycles, throughput). The obtained results of original HIGHT implementation are compared against a similar study. The modified HIGHT algorithm suggested in this study brings out augmented security with trifling drop off in unique attributes of embedded devices, the cost and performance.

**Key words:** Block cipher, light weight cryptography, HIGHT algorithm, AVR, 8 bit microcontroller, embedded security

### INTRODUCTION

Exchange of information between a sender and a receiver is no longer secured when communicated in an open medium. The possible intruders in the communication channel always works to diminish the confidentiality of the private data (Liu *et al.*, 2011; Rabah, 2005a, b). Making the data invisible from the eyes of the intruders, are suggested by experts with diverse schemes of steganography (Hmood *et al.*, 2010; Amirtharajan *et al.*, 2012; Amirtharajan and Rayappan, 2013; Amirtharajan *et al.*, 2013a-j; Praveenkumar *et al.*, 2014a-l). Cryptographic techniques (Al-Somani *et al.*, 2006, 2009; Jaber *et al.*, 2012; Salem *et al.*, 2011) have been suggested to make the visible data in a shared channel secured from unintended receiver.

Crypto algorithms (Schneier, 2007) do substitution, transpositioning and transformation mechanisms involving a key in order to formulate the secret data in an unreadable form from the perspective of inadvertent user's. The kind of key used divides it into two categories namely symmetric (private) (Rabah, 2005c; Abomhara *et al.*, 2010; Muda *et al.*, 2010) and asymmetric (public) (Mousa, 2005; Rabah, 2006; Wang *et al.*, 2007) algorithms. Symmetric key algorithms that

takes a sequence of bits as input and encrypts the plain text bit by bit is called stream ciphers whereas, when it takes a block of bytes as input and encrypts a block of data at a time is known as block ciphers (Cakiroglu, 2010). These block ciphers are used in many application areas like banking, etc. The key size and number of rounds are the chief factor deciding the security level of crypto algorithms (Zaidan *et al.*, 2010).

Although, several crypto algorithms exists, the special kind Light Weight Cryptography (LWC) (Eisenbarth *et al.*, 2007, 2012; Sadanandan and Mahalingam, 2009) suits well for the resource constrained devices (Rinne *et al.*, 2007) serving the purpose of computational engine in WSNs. LWC is aimed at reduction in various factors based on the kind of target used for implementation. For fully customized hardware implementation on ASIC, the requirement on number of logic gates must be minimized. This sort of implementation provides greatest level of power and size reduction at the cost of rise in Non-Recurring Engineering (NRE) cost.

The arrival of Field Programmable Gate Arrays (FPGAs) (Rajagopalan *et al.*, 2012; Yalla and Kaps, 2009) brings down the NRE cost with minimized power consumption and development time. Further cutback in cost can be achieved with semi custom devices like microcontrollers (Standaert *et al.*, 2006). These devices come with two different architectures namely Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computers (RISC). Both architectures contain less amount of memory storage and can be operated under optimal frequency ranges. Among them, RISC architectures are popular for its low power operation with high performance and throughput measured by Million Instructions Per Second (MIPS).

Atmega series microcontrollers from AVR family developed by ATMEL are the popular one widely used in Wireless Sensor Nodes (WSNs). WSNs enclose sensor devices and communication module integrated with computing facility provided by microcontrollers. These WSNs are deployed in several discrete points for data collection and they tend to communicate their data to a central node. The security, during this data communication, can be accomplished by cryptography algorithms. HIGHT, PRESENT and Humming Bird are some of the examples for light weight algorithms which are suitable for implementation with microcontrollers (Janakiraman *et al.*, 2012). ATmega8 is a popular 8 bit RSIC architecture that contains on-chip data and program storage of 1 and 8 kB, respectively. It is capable of running at a maximum frequency of 16MHz at which it can provide a throughput upto 16 MIPS. Its on-chip peripheral modules include Analog to Digital Converter (ADC) with 10 bit resolution and serial communication devices (Twin Wire Interface (TWI), Serial Peripheral Interface (SPI) and Universal Synchronous Asynchronous Receiver Transmitter (USART)) that satisfy the requirement for WSN connectivity. The HIGHT (Hong *et al.*, 2006) is a block cipher algorithm that takes 64 bit plaintext and 128 bit symmetric key as input and generates 64 bit cipher text as output. As the algorithm uses only 8 bit XOR, bit wise rotate and addition based on modulo  $2^8$  operations, the 8 bit microcontroller is the finest target for implementation.

This study provides a simple yet an ingenious way to encrypt information in under 8 bit microcontroller.

## **MATERIALS AND METHODS**

The performance of various LWC algorithms on different microcontroller platforms and its implementation efficiency based on execution cycles and memory foot print have been analyzed by

many researchers. Janakiraman *et al.* (2012) showed a method to randomize the sequence of sub keys selection with a 4 bit Linear Feedback Shift Register (LFSR) (Sundararaman and Upadhyay, 2011) used in the existing Humming bird algorithm (Janakiraman *et al.*, 2014) to raise its security level. Humming bird is a 16 bit block cipher that uses a 256 bit symmetric key and a Substitution Box (S-box) to encrypt the data with 16 rounds. LFSR can be used to generate pseudo random numbers (Rajagopalan *et al.*, 2014) based on the number of stages used.

The non-existence of S-box in HIGHT algorithm minimizes the claim on memory size of the microcontrollers. In this study, the HIGHT algorithm is implemented on AVR ATmega8 RISC microcontroller. The algorithm consists of 32 round functions with first round preceded by an initial transformation and last round succeeded by a final transformation. The 128 bit original key is used to produce eight; byte sized whitening keys in order to provide four different keys to be used by initial and final transformations each. Subsequently, four 8 bit sub keys are generated to satisfy the unique key requirement for each of the 32 rounds. All the keys, plain text and cipher text are stored in the internal SRAM of ATmega8 microcontroller.

The overall encryption process is shown in Fig. 1. The decryption process of HIGHT algorithm generates the whitening keys and sub keys using the procedure similar to the one used for encryption. The three transformation process namely initial transformation, round functions and final transformation uses the reverse procedure with same number of rounds as in encryption to obtain the original plain text as output.

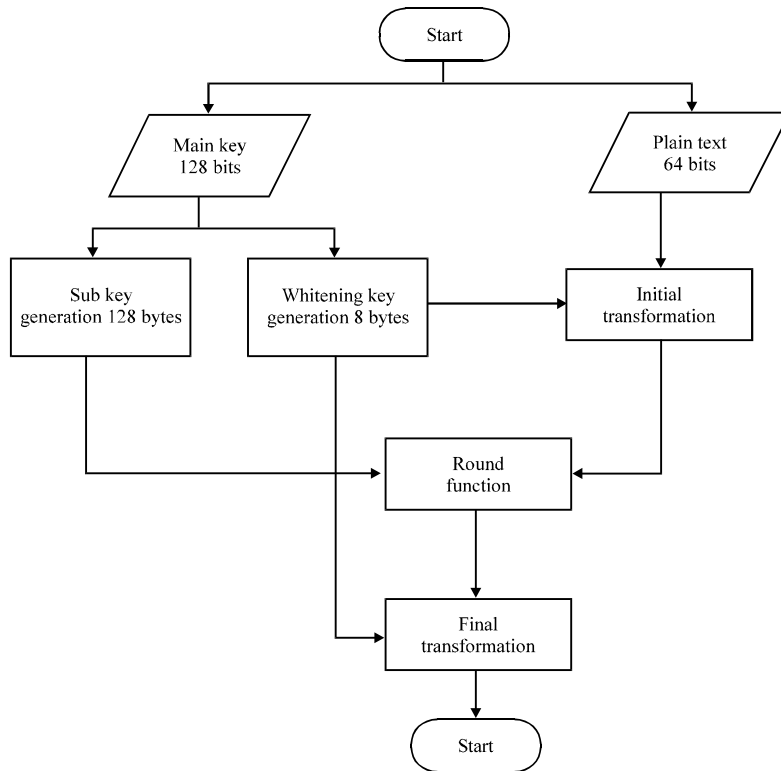
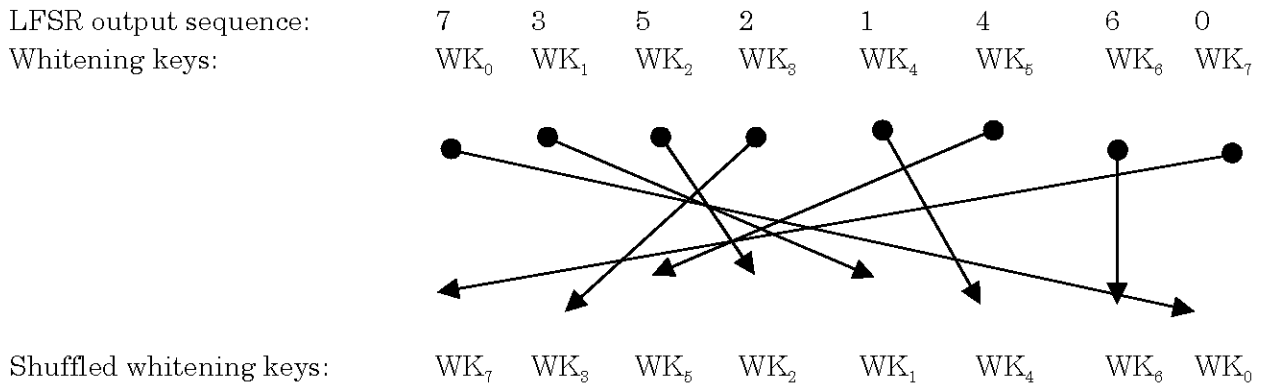


Fig. 1: Flow Chart of HIGHT algorithm

**METHODOLOGY**

**Key shuffle and plaintext pre processing:** To improve the security level of the HIGHT algorithm, a 3 bit LFSR as shown in Fig. 2 is newly introduced in it. An ‘n’ bit LFSR always produces results for  $2^n-1$  stages. This 3 bit LFSR circuit generates a seven stage sequence in a pseudo random fashion based on its initial state (seed value) as mentioned in the Table 1.

The LFSR output is used by two new functions that are introduced to make intended modifications in the actual algorithm flow to provide additional level of security. The original HIGHT algorithm uses the first four out of the eight whitening keys ( $WK_{0,7}$ ) for the initial transformation and the last four for the final transformation. The modified HIGHT proposed in this study uses a function that shuffles the order of eight whitening keys as per the 7 stage sequence output of the 3 bit LFSR. A sample key shuffling process is shown as follows:



**HIGHT algorithm:**

Keys for initial transformation:  $WK_0$   $WK_1$   $WK_2$   $WK_3$   
 Keys for final transformation:  $WK_4$   $WK_5$   $WK_6$   $WK_7$

**Modified HIGHT algorithm:**

Keys for initial transformation:  $WK_7$   $WK_3$   $WK_5$   $WK_2$   
 Keys for final transformation:  $WK_1$   $WK_4$   $WK_6$   $WK_0$

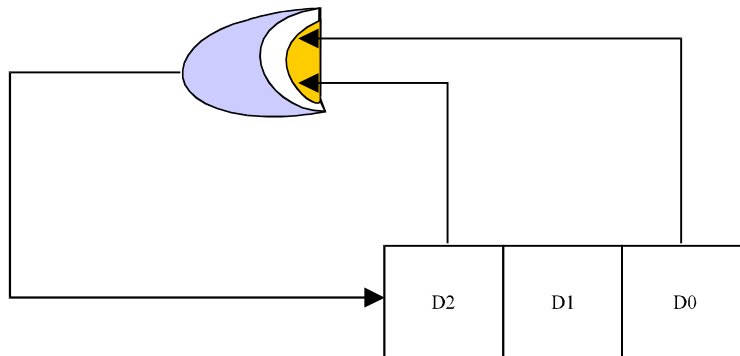
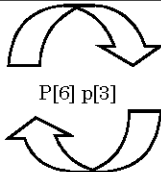
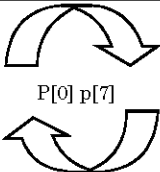


Fig. 2: The 3 bit LFSR with EX-OR feedback

Table 1: Output sequence of 3 bit LFSR

D2	D1	D0	Stage value (seed)
1	1	1	
0	1	1	3
1	0	1	5
0	1	0	2
0	0	1	1
1	0	0	4
1	1	0	6
0	0	0	0 ← Manually inserted

Table 2: Plain text pre-process

LFSR[6] = 0	LFSR[6] = 3	LFSR[6] = 5	LFSR[6] = 6
$p[1] = p[1] \text{ XOR } (p[2] \text{ OR LFSR}[6])$	$p[4] = p[4] \text{ XOR } (p[5] \text{ OR LFSR}[6])$	 <p>P[6] p[3]</p>	 <p>P[0] p[7]</p>

A simple manipulation on the plain text information is carried out in the modified HIGHT algorithm before the start of initial transformation. As this module processes the plain text prior to the beginning of plain text processing by the actual algorithm, the job of this module is termed as plain text pre-processing. Here, either one byte value modification or position swapping of two bytes of plaintext may occur with a probability of 0.5 based on the last stage value of 3 bit LFSR. On the remaining probability of 0.5, any change may not occur in plaintext. The plain text pre process operation is described in Table 2.

The pseudo code for the encryption of modified HIGHT algorithm is given as follows.

Pseudo code for modified HIGHT encryption

```

For (each 64 bit plain text block, P)
{
Get 128 bit Original Key, mk;
unsigned char Random (8), LFSR_seed;
for (i = 0; i < 16; i++)
{
Random[i] = LFSR_3 bit (LFSR_seed);
}
WK = Whitening Key Generation (mk);
SWK = Whitening Key Shuffle (Random);
Subkey Generation (mk);
PTP = Plain Text Pre Process (P);
X = Initial Transformation (PTP,SWK);
for(i = 1; i = 32; i++)
{
Zi = Round Function (X);
}
64 bit Cipher Text, C = final Transformation (Z32,SWK);
}
    
```

The operations performed by the newly added functions whitening Key Shuffle and Plain Text Pre Process, the point of the call for these added functions along with their preceding and succeeding functions are mentioned in the pseudo code for encryption.

The coding for the HIGHT encryption/decryption with and without the proposed added functions is written in embedded C language for AVR ATmega8 microcontroller. The coding was simulated using the open source tool Atmel Studio 6 which has the integrated compilers AVR GCC and ARM GCC for AVR and ARM family of devices.

## RESULTS AND DISCUSSION

All the results in terms of code size and performance analysis in terms of execution time are obtained from the debug options provided in the Atmel Studio. The raise in security level of the modified HIGHT algorithm in comparison with the original one is obvious due to the additional randomization done on whitening key sequence and plain text manipulation done prior to the initial round. In order to find the effectiveness of this modified algorithm implementation against the original one, the parameters code size, execution cycles and throughput are analyzed in this session. The optimization features of the AVR GCC compiler are used to find the impact of optimization on code size and execution time. The O3 level of optimization which can produce the fastest code at the cost of demanding more number of bytes in FLASH memory is analyzed and the values for encryption/decryption of HIGHT and Modified HIGHT are tabulated in Table 3. On the other side, Os level of optimization which is mainly to optimize for code size that also gives good optimization in all aspects. Table 4 gives this code optimization results.

The overhead in terms of FLASH memory bytes on the encryption and decryption process of the modified HIGHT algorithm when compared with original HIGHT algorithm is shown in Fig. 3 and 4. Although, around 25-30% of code rise is reported in modified HIGHT, the overall code memory occupied by these algorithms on the existing 8KB FLASH of ATmega8 is only less than 40% even in time optimized version that produced larger code.

Table 3: HIGHT vs. Modified HIGHT-time optimized

Algorithm	Memory footprint (bytes)	Execution cycle count
<b>HIGHT</b>		
Encryption	2372	11163
Decryption	2560	14265
<b>Modified HIGHT</b>		
Encryption	3098	11388
Decryption	3224	14487

Table 4: HIGHT vs. modified HIGHT-code optimized

Algorithm	Memory footprint (bytes)	Execution cycle count
<b>HIGHT</b>		
Encryption	872	22677
Decryption	1012	24070
<b>Modified HIGHT</b>		
Encryption	1086	22992
Decryption	1228	24381

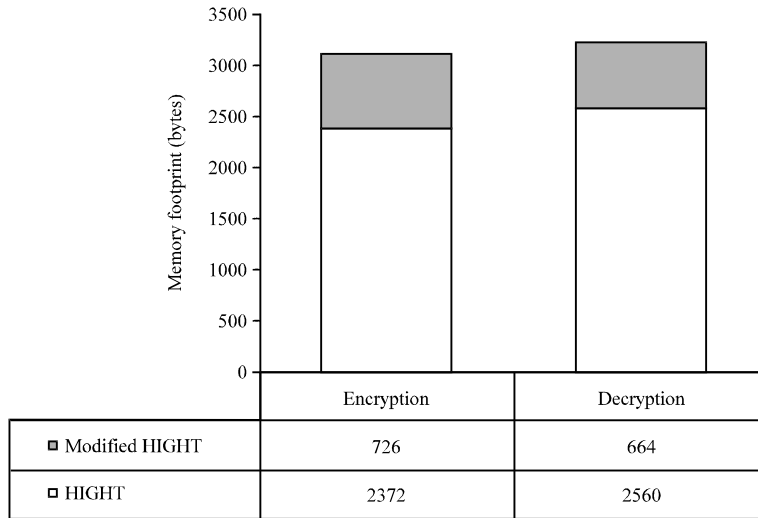


Fig. 3: Memory overhead in modified HIGHT-time optimized

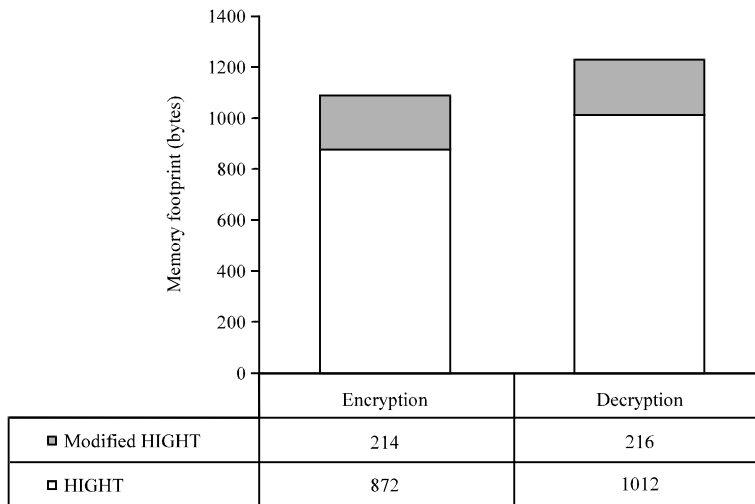


Fig. 4: Memory overhead in modified HIGHT-code optimized

In contrast to the overhead raised in code size of the modified algorithm, Fig. 5 and 6 shows the fee paid in terms of timing overhead is only negligible for the raise in security with added functions of the modified algorithm.

To justify the efficiency of our C code and the advanced AVR GCC compiler of Atmel studio 6, the results of the original code implemented in this study are compared with the earlier results reported by Eisenbarth *et al.* (2012). Table 5 and 6 gives the performance comparison between the results by Eisenbarth *et al.* (2012) and the results obtained in this study for encryption and decryption algorithms respectively. In the Table 5 and 6 the highlighted rows of column one gives the results of algorithms implemented in this study. The time optimized execution cycle counts given in Table 3 are duplicated in column 3 of Table 5 and 6. The fourth column of the Table 5 and 6 gives the number cycles taken by the implementations to encrypt a bit of plaintext including the time for initialization and key generations.



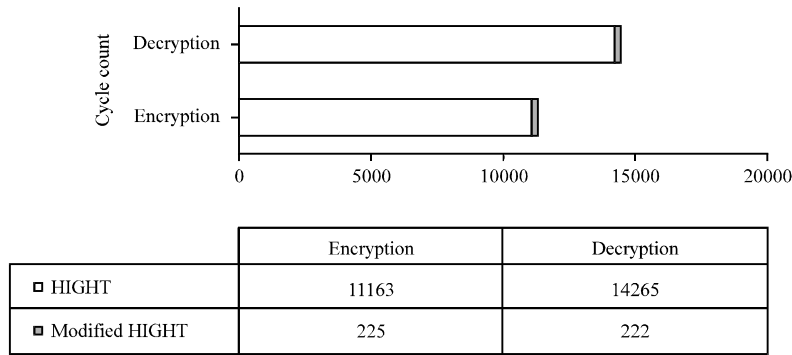


Fig. 5: Timing overhead in modified HIGHT-time optimized

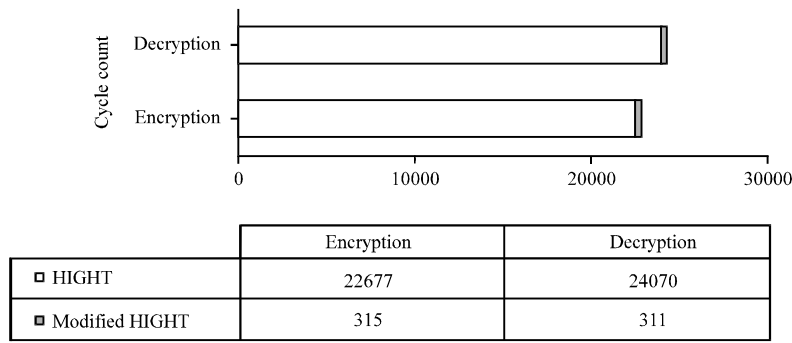


Fig. 6: Timing overhead in modified HIGHT-code optimized

Table 5: Performance comparison-encryption

Implementation	Block size (bit)	Encryption cycles (cycles)	Encryption cycles (cycles bit <sup>-1</sup> )	Throughput (bits sec <sup>-1</sup> )
Eisenbarth <i>et al.</i> (2012)	64	19503	304.73	52505.50
HIGHT	64	11163	174.42	91732.60
Modified HIGHT	64	11388	177.93	89923.00

Table 6: Performance comparison-decryption

Implementation	Block size (bit)	Decryption cycles (cycles)	Decryption cycles (cycles bit <sup>-1</sup> )	Throughput (bits sec <sup>-1</sup> )
Eisenbarth <i>et al.</i> (2012)	64	20159	314.98	50796.87
HIGHT	64	14265	222.89	71784.29
Modified HIGHT	64	14487	226.36	70683.87

Based on the execution cycles for HIGHT encryption and decryption reported by Eisenbarth *et al.* (2012), the throughput values are calculated assuming 16 MHz operating frequency and indicated in the last column of the Table 5 and 6. Throughput of our implementation is raised considerably in comparison with Eisenbarth *et al.* (2012). It is also proven from Fig. 7 and 8, the inclusion of new functions in the modified HIGHT algorithm does not make considerable dip in the throughput.

The sample values taken as 64 bit Plaintext and 128 bit symmetric key; the corresponding Ciphertext produced by both HIGHT and Modified HIGHT algorithm in ASCII and Hexadecimal forms are shown as follows.

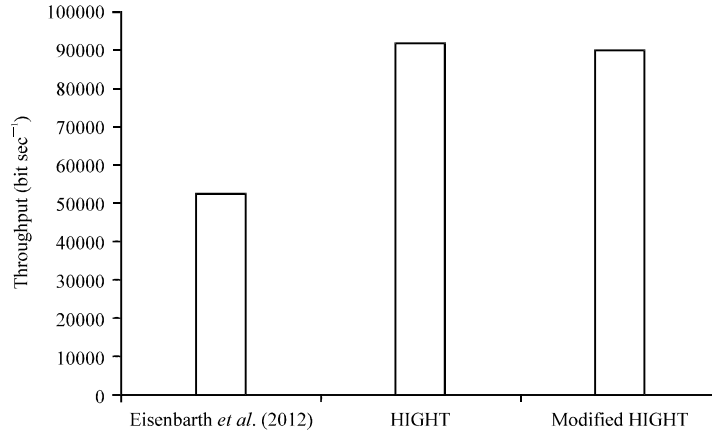


Fig. 7: Throughput comparison-encryption

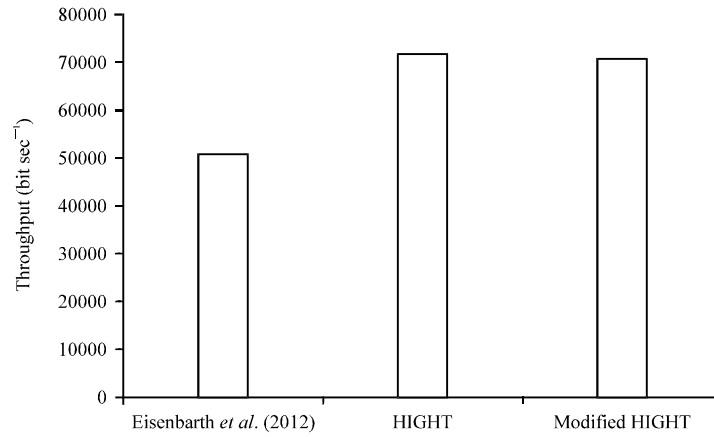


Fig. 8: Throughput comparison-decryption

**Sample input**

Plain text: HIGHTenc  
 [0x48 0x49 0x47 0x48 0x54 0x65 0x6E 0x63]  
 Key: KE4MODIFIEDhight  
 [0x4B 0x45 0x34 0x4D 0x4F 0x44 0x49 0x46 0x49 0x45 0x44 0x68  
 0x69 0x67 0x68 0x74]

**Sample output-HIGHT**

Cipher text: "OIí:Íáf  
 (HIGHT) [0xA8 0x4F 0x49 0xED 0x3A 0xCD 0xE2 0x83]

**Sample output-modified HIGHT**

Pre processed plain text: cIGHTenH  
 [0x63 0x49 0x47 0x48 0x54 0x65 0x6E 0x48]  
 Cipher text: Æ..Bù = Ò  
 (Modified HIGHT) [0x04 0x8C 0x0B 0xBC 0x42 0xF9 0x3D 0xD2]

## CONCLUSION

The above comparison charts proves that the embedded C code for HIGHT algorithm optimized with AVR GCC compiler of Atmel Studio 6 for ATmega8 can perform better than the prior implementation reported in Eisenbarth *et al.* (2012). This improvement in terms of execution cycles and throughput are achieved through time optimization resulted at the cost of memory size. In addition, the enhancement in security level through the additional 3 bit LFSR and other additional functionalities proposed in the modified HIGHT algorithm of this study produces performance metrics that are competing with the original algorithm.

## ACKNOWLEDGMENT

The authors wish to acknowledge SASTRA University for providing infrastructural support to carry out this study.

## REFERENCES

- Abomhara, M., O.O. Khalifa, O. Zakaria, A.A. Zaidan, B.B. Zaidan and H.O. Alanazi, 2010. Suitability of using symmetric key to secure multimedia data: An overview. *J. Applied Sci.*, 10: 1656-1661.
- Al-Somani, T.F., E.A. Khan, A.M. Qamar-ul-Islam and H. Houssain, 2009. Hardware/software co-design implementations of elliptic curve cryptosystems. *Inform. Technol. J.*, 8: 403-410.
- Al-Somani, T.F., M.K. Ibrahim and A. Gutub, 2006. High performance elliptic curve GF(2<sup>m</sup>) crypto-processor. *Inform. Technol. J.*, 5: 742-748.
- Amirtharajan, R., J. Qin and J.B.B. Rayappan, 2012. Random image steganography and steganalysis: Present status and future directions. *Inform. Technol. J.*, 11: 566-576.
- Amirtharajan, R. and J.B.B. Rayappan, 2013. Steganography-time to time: A review. *Res. J. Inform. Technol.*, 5: 53-66.
- Amirtharajan, R., K. Karthikeyan, M. Malleswaran and J.B.B. Rayappan, 2013a. Kubera kolam: A way for random image steganography. *Res. J. Inform. Technol.*, 5: 304-316.
- Amirtharajan, R., M.V. Abhiram, G. Revathi, J.B. Reddy, V. Thanikaiselvan and J.B.B. Rayappan, 2013b. Rubik's cube: A way for random image steganography. *Res. J. Inform. Technol.*, 5: 329-340.
- Amirtharajan, R., P. Archana and J.B.B. Rayappan, 2013c. Why image encryption for better steganography. *Res. J. Inform. Technol.*, 5: 341-351.
- Amirtharajan, R., S. Sulthana and J.B.B. Rayappan, 2013d. Seeing and believing is a threat: A visual cryptography schemes. *Res. J. Inform. Technol.*, 5: 435-441.
- Amirtharajan, R., K.M. Ashfaq, A.K. Infant and J.B.B. Rayappan, 2013e. High performance pixel indicator for colour image steganography. *Res. J. Inform. Technol.*, 5: 277-290.
- Amirtharajan, R., R. Subrahmanyam, J.N. Teja, K.M. Reddy and J.B.B. Rayappan, 2013f. Pixel indicated triple layer: A way for random image steganography. *Res. J. Inform. Technol.*, 5: 87-99.
- Amirtharajan, R., S.D. Roy, N. Nesakumar, M. Chandrasekar, R. Sridevi and J.B.B. Rayappan, 2013g. Mind game for cover steganography: A refuge. *Res. J. Inform. Technol.*, 5: 137-148.
- Amirtharajan, R., V. Rajesh, P. Archana and J.B.B. Rayappan, 2013h. Pixel indicates, standard deviates: A way for random image steganography. *Res. J. Inform. Technol.*, 5: 383-392.
- Amirtharajan, R., P.S. Priya and J.B.B. Rayappan, 2013i. Pixel indicated user indicator: A muxed stego. *Res. J. Inform. Technol.*, 5: 73-86.

- Amirtharajan, R., G. Devipriya, V. Thanikaiselvan and J.B.B. Rayappan, 2013j. High capacity triple plane embedding: A colour stego. *Res. J. Inform. Technol.*, 5: 373-382.
- Cakiroglu, M., 2010. Software implementation and performance comparison of popular block ciphers on 8-bit low-cost microcontroller. *Int. J. Phys. Sci.*, 5: 1338-1343.
- Eisenbarth, T., S. Kumar, C. Paar, A. Poschmann and L. Uhsadel, 2007. A survey of lightweight-cryptography implementations. *IEEE Design Test Comput.*, 24: 522-533.
- Eisenbarth, T., Z. Gong, T. Guneyusu, S. Heyse and S. Indesteege *et al.*, 2012. Compact implementation and performance evaluation of block ciphers in ATtiny devices. *Proceedings of the 5th International Conference on Cryptology*, July 10-12, 2012, Africa, Ifrance, Morocco, pp: 172-187.
- Hmood, A.K., H.A. Jalab, Z.M. Kasirun, B.B. Zaidan and A.A. Zaidan, 2010. On the capacity and security of steganography approaches: An overview. *J. Applied Sci.*, 10: 1825-1833.
- Hong, D., J. Sung, S. Hong, J. Lim and S. Lee *et al.*, 2006. HIGHT: A new block cipher suitable for low-resource device. *Cryptogr. Hardware Embedded Syst., LNCS*, 4249: 46-59.
- Jaberi, A., R. Ayanzadeh and A.S.Z. Mousavi, 2012. Two-layer cellular automata based cryptography. *Trends Applied Sci. Res.*, 7: 68-77.
- Janakiraman, S., K.V.S.K. Kumar, R.R.K. Reddy, A. Srinivasulu, R. Amirtharajan, K. Thenmozhi and J.B.B. Rayappan, 2014. Humming bird with coloured wings: A feedback security approach. *Inform. Technol. J.*, 13: 2022-2026.
- Janakiraman, S., R. Amirtharajan, K. Thenmozhi and J.B.B. Rayappan, 2012. Firmware for data security: A review. *Res. J. Inform. Technol.*, 4: 61-72.
- Liu, C., Y. Zhou, Y. Xiao and G. Sun, 2011. Encryption algorithm of RSH (round sheep hash). *Inform. Technol. J.*, 10: 686-690.
- Mousa, A., 2005. Sensitivity of changing the RSA parameters on the complexity and performance of the algorithm. *J. Applied Sci.*, 5: 60-63.
- Muda, Z., R. Mahmud and M.R. Sulong, 2010. Key transformation approach for rijndael security. *Inform. Technol. J.*, 9: 290-297.
- Praveenkumar, P., R. Amirtharajan, K. Thenmozhi and J.B.B. Rayappan, 2014a. Sub carriers carry secret: An absolute stego approach. *J. Applied Sci.*, 14: 1728-1735.
- Praveenkumar, P., R. Amirtharajan, K. Thenmozhi and J.B.B. Rayappan, 2014b. Double layer encoded encrypted data on multicarrier channel. *J. Applied Sci.*, 14: 1689-1700.
- Praveenkumar, P., K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014c. Purposeful error on OFDM: A secret channel. *Inform. Technol. J.*, 13: 1985-1991.
- Praveenkumar, P., G.S. Hemalatha, B. Reddy, K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014d. Secret link through simulink: A stego on OFDM channel. *Inform. Technol. J.*, 13: 1999-2004.
- Praveenkumar, P., K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014e. Stego in multicarrier: A phase hidden communication. *Inform. Technol. J.*, 13: 2011-2016.
- Praveenkumar, P., K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014f. Inserted embedding in OFDM channel: A multicarrier stego. *Inform. Technol. J.*, 13: 2017-2021.
- Praveenkumar, P., K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014g. Data puncturing in OFDM channel: A multicarrier stego. *Inform. Technol. J.*, 13: 2037-2041.
- Praveenkumar, P., R. Deepak, K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014h. Reversible steganography on OFDM channel: A role of cyclic codes. *Inform. Technol. J.*, 13: 2047-2051.

- Praveenkumar, P., K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014i. Reversible steganography on OFDM channel-a role of RS coding. *Inform. Technol. J.*, 13: 2052-2056.
- Praveenkumar, P., K. Thenmozhi, J.B.B. Rayappan and R. Amirtharajan, 2014j. Spread and hide-a stego transceiver. *Inform. Technol. J.*, 13: 2061-2064.
- Praveenkumar, P., R. Amirtharajan, K. Thenmozhi and J.B.B. Rayappan, 2014k. Coded crypted converted hiding (C<sup>3</sup>H)-a stego channel. *J. Applied Sci.*, 14: 1786-1797.
- Praveenkumar, P., R. Amirtharajan, R.S. Janani, K. Thenmozhi and J.B.B. Rayappan, 2014l. Multi (Carrier+Modulator) adaptive system-an anti fading stego approach. *J. Applied Sci.*, 14: 1836-1843.
- Rabah, K., 2005a. Theory and implementation of elliptic curve cryptography. *J. Applied Sci.*, 5: 604-633.
- Rabah, K., 2005b. Secure implementation of message digest, authentication and digital signature. *Inform. Technol. J.*, 4: 204-221.
- Rabah, K., 2005c. Theory and implementation of data encryption standard: A review. *Inform. Technol. J.*, 4: 307-325.
- Rabah, K., 2006. Implementing secure RSA cryptosystems using your own cryptographic JCE provider. *J. Applied Sci.*, 6: 482-510.
- Rajagopalan, S., R. Amirtharajan, H.N. Upadhyay and J.B.B. Rayappan, 2012. Survey and analysis of hardware cryptographic and steganographic systems on FPGA. *J. Applied Sci.*, 12: 201-210.
- Rajagopalan, S., Y. Ravishankar, H.N. Upadhyay, J.B.B. Rayappan and R. Amirtharajan, 2014. Modeling combo PR Generator for Stego Storage Self Test (SSST). *Inform. Technol. J.*, 13: 1936-1944.
- Rinne, S., T. Eisenbarth and C. Paar, 2007. Performance analysis of contemporary light-weight block ciphers on 8-bit microcontrollers. *SPEED 2007*, [http://www.ei.rub.de/media/crypto/veroeffentlichungen/2011/01/29/lw\\_speed2007.pdf](http://www.ei.rub.de/media/crypto/veroeffentlichungen/2011/01/29/lw_speed2007.pdf)
- Sadanandan, S. and R. Mahalingam, 2009. Light Weight Cryptography and Applications. In: *Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics*, Sobh, T., K. Elleithy and A. Mahmood (Eds.). Springer, Netherlands, ISBN: 9781402087363, pp: 484-488.
- Salem, Y., M. Abomhara, O.O. Khalifa, A.A. Zaidan and B.B. Zaidan, 2011. A review on multimedia communications cryptography. *Res. J. Inform. Technol.*, 3: 146-152.
- Schneier, B., 2007. *Applied Cryptography: Protocols, Algorithm and Source Code in C*. 2nd Edn., John Wiley and Sons, New Delhi, India.
- Standaert, F.X., G. Piret, N. Gershenfeld and J.J. Quisquater, 2006. SEA: A scalable encryption algorithm for small embedded applications. *Smart Card Res. Adv. Appl.*, 3928: 222-236.
- Sundararaman, R. and H.N. Upadhyay, 2011. Stego system on chip with LFSR based information hiding approach. *Int. J. Comput. Appl.*, 18: 24-31.
- Wang, L., H. Zhao and G. Bai, 2007. A cost-efficient implementation of public-key cryptography on embedded systems. *Proceedings of the 2007 International Workshop on Electron Devices and Semiconductor Technology*, June 3-4, 2007, Tsinghua University, pp: 194-197.
- Yalla, P. and J. Kaps, 2009. Lightweight cryptography for FPGAs. *Proceedings of the International Conference on ReConfigurable Computing and FPGAs*, December 9-11, 2009, Quintana Roo, pp: 225-230.
- Zaidan, B.B., A.A. Zaidan, A.K. Al-Frajat and H.A. Jalab, 2010. On the differences between hiding information and cryptography techniques: An overview. *J. Applied Sci.*, 10: 1650-1655.