http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 13 (12): 2022-2026, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.2022.2026

Humming Bird with Coloured Wings: A Feedback Security Approach

Siva Janakiraman, K.V. Sai Krishna Kumar, R. Rohit Kumar Reddy, A. Srinivasulu,
Rengarajan Amirtharajan, K. Thenmozhi and John Bosco Balaguru Rayappan
School of Electrical and Electronics Engineering, SASTRA University, 613401, India

Abstract: The day to day amplification on technology and the demand on shrinking the device sizes reflect in the diminution on memory sizes and constraints on device outlay. In the present times of mass communication with the swift escalating technology, there is a dire necessity for security on the information traversing between authorized individuals. All these factors invite the property of light weightiness and the focused security algorithms such as Hummingbird for devices with humiliated resources. This study present an efficient software implementation of modified Humming Bird algorithm on an 8-bit microcontroller with randomization on sub-key selection based on LFSR. We also do comparative analysis on performance in terms of memory foot print and execution time between our modified one and the original algorithm implementations. The obtained results make it evident that the proposed method raises the complexity significantly without hike in resource consumption and makes it more suitable to provide security in extremely constrained devices.

Key words: Data security, cryptography, LFSR, hummingbird, light weight cryptography

INTRODUCTION

In the present scenario of widely evolving technologies of communication and hence, the information security of critical data arises with the requirement for impregnable covert channel arises when critical information has to be communed. Modern cryptographic methods use a key to control encryption and decryption (Schneier, 2007). Two classes of key-based encryption algorithms are namely symmetric (secret-key) and asymmetric (public-key) (Abomhara *et al*., 2010; Muda *et al*., 2010). Symmetric ciphers can be divided into stream ciphers (encrypt a single bit of plaintext at a time) and block ciphers (take a number of bits and encrypt them as a single unit) (Roman*et al*., 2007). In symmetric algorithms, both sender and receiver share the same key to encrypt as well as to decrypt the messages. In order to provide privacy and thus, security, this key needs to be kept covert. The amount of power consumed by the symmetric algorithms for computation is always much less. Among the well-known, a few like: DES, AES, Triple-DES (Zaidan *et al*., 2010a; Leander *et al*., 2007; Daemen and Rijmen, 2002), BLOWFISH (symmetric 64-bit block cipher. Key lengths can vary from 32 to 448 bits in length) and TWOFISH (128-bit block cipher using 128-, 192-, or 256-bit keys). DES better known the Digital Encryption Standard is a widely used symmetric algorithm which encrypts a 64-bit block using a 56-bit key (Rabah, 2005a). However, asymmetric algorithms implement the use of a pairs of keys-one for encryption and the other for decryption. While a decryption key is basically kept a secret and hence called the "private key" or "secret key", the encryption key is passed to everyone who ever wants to encrypt and send messages, so called "public key". The reconstruction of secret key is not possible using public key. The motivation of asymmetric algorithms was published by Diffie and Hellmann (1976). The Asymmetric algorithms turned to be the best for real-world implementation since the secret key remains clandestine and thus, the threat of getting recognized is much lesser. Some of the well-known asymmetric algorithms are RSA (Gura *et al*., 2004; Mousa, 2005), DSA. Due to the property of slowness in asymmetric algorithms compared to symmetric ones, in many applications, an amalgam of both is being used. In vision of this, the use of asymmetric keys to provide authentication (Rabah, 2005b) and later than the successful implementation; one or more of the symmetric keys are evolved and exchanged using the former i.e., asymmetric encryption. On a further note, the advantages of both algorithms can be utilized to overcome the limitations. Classic examples for this procedure are RSA/IDEA (International Data Encryption Algorithm) combination of PGP2. In the current digital era,

Corresponding Author: Siva Janakiraman, School of Electrical and Electronics Engineering, SASTRA University, 613401, India

the Implementation of cryptography or steganography algorithms on application specific or reconfigurable hardware platforms has become popular. Always the cost effectiveness on microcontrollers and faster performance on FPGAs being the weapon in the battle field for device selection (Rajagopalan *et al*., 2012a, b; Zaidan *et al*., 2010b). As goes the quotation "As light as a feather and as hard as dragon scales", there evolves the 'light-weight cryptography' (Rinne *et al*., 2007 and Janakiraman *et al*., 2012) in the realm of information security with lightweight algorithms involving a short internal state (to lower area) to allow serialization (to lower power). However, the characteristic feature remains to be their short processing time (to lower energy) and short output (to lower communication cost). Thus, the cryptography is well-suited for (extremely) constrained devices and not purposed for all-powerful adversaries. Though, it serves to be a strong crypto, it is no comparison to traditional cryptography. Therefore, it is known to be a pervasive methodology which primarily is an amalgam of wireless, embedded which results in a constrained CPU, memory and finally at the scope of battery (Janakiraman *et al*., 2012; Salem *et al*., 2011). In this work we suggest a method to perk up the security level of the Humming bird algorithm and also we analyze its impact on memory occupancy and execution speed in comparison with original algorithm (Engels *et al*., 2010).

Humming Bird is a well known light weight symmetric cipher that was designed with a focus for constrained devices (Engels *et al*., 2010). The algorithm has the following specifications:

- Size of a plain text block is 16 bits
- Five 16 bit registers RS_i (i = 1 to 4) and a 16 bit Linear Feedback Shift Register (LFSR) are used
- A symmetric key of size 256 bit is used, which is divided into 16 sub keys of 16 bits each

The algorithm consists of the following three main processes.

Initialization process: The process starts by choosing four 16 bit values that are used to initialize the four internal state registers RS_i (i = 1 to 4) and then performing modulo addition operation between the registers before the process of encryption.

Encryption process: This is same as initialization process where the input is a 16-bit plain text (PT_i) and the output of cipher block (Ek4) is the cipher text (CT_i).

Cipher block (Eki): Each cipher block has a block size of 16 bits and uses a 64 bit key (K_i). This block has four regular rounds and a final round. The 64 bit key is sub-divided into four 16-bit keys (ki,1., ki,2., ki,3., ki,4) which will be used in four regular rounds. The final round uses two derived keys ki,5 where, ki,5 = (ki,1^ki,3) and ki,6 = (ki,2^ki,4). Each round has the following three steps:

- Exclusive-OR operation of input and sub key ki,j
- Mapping the obtained result using S-box
- Applying linear transform L (m) = circular Left shift for 7 bits (m) for the S-box result

The result obtained after 4 rounds is EX-ORed with derived key k5 and the result obtained is mapped using S-box. Then Exclusive-OR operation is performed with derived key k6. The result obtained is the output of the cipher block (Ek_i).

Decryption process: This is similar to the encryption process. Here the registers and cipher blocks are taken in reverse order to that of encryption process and modulo 2^{16} subtraction is performed instead of modulo addition. The cipher text (CT_i) obtained in encryption process is given as the input to the decryption process to obtain the plain text (PT_i) as decrypted output.

Inorder to provide more security randomization can be done at various levels. Rajagopalan and Upadhyay (2011), described the implementations on generation of pseudo-random numbers using LFSR. Based on the results obtained on software implementations and analysis of Light weight crypto systems made by Rinne *et al.* (2007) hummingbird gives better performance on both during size and code optimizations on comparison with other well known light weight algorithm present.

PROPOSED METHOD

Random sub-key selection (RSS) using LFSR: The two main objectives of a good symmetric crpto algorithm are:

- The algorithm itself should be known, only the key is secret
- Key should be randomized with a good algorithm in such a way to make identification of key is impossible though the plaintext and cipher-text are known

In order to improve the complexity level of the humming bird algorithm, we propose a random sequence

for the usage of the 16 sub-keys with the help of a 4-bit LFSR. The LFSR takes a 4-bit initial value as its seed as shown in Fig. 1 to produce random numbers between 1 to 15 and finally a '0' is appended as the 16th value to select the usage sequence 0 to 15 of 16-subkeys used to encrypt or decrypt a plaintext block. The operation of 4-bit LFSR is shown in the Fig. 2.

The following process shows the sample calculation for random number generation using LFSR Let us take:

L.F.S.R = [1 0 1 0 10 1 0 0 1 1 1 0 1 1 1]

So by taking bits (0 1 1 1) as seed values to the shift registers we get the results as given in Table 1. This brings out the random numbers as:

Rand [16] = [7,3,1,8,4,2,12,6,11,5,10,13,14,15,0]

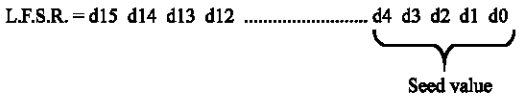

Fig. 1: Seed value selection

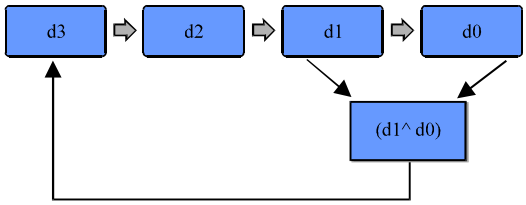

Fig. 2: Operation of 4-bit LFSR

Table 1: Stage by stage shifted output of 4-bit LFSR

	Binary Output of 4-bit LFSR					
Stage No.	d3	d2	d1	d0	Decimal equivalent	d1^d0
1	0	1	1	1	7	0
2	0	0	1	1	3	0
3	0	0	0	1	1	1
4	1	0	0	0	8	0
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
15	1	1	1	1	15	0

Note: the 16th digit '0' was inserted manually after the generation of random numbers 1 to 15 by LFSR. The pseudo code given below describes the procedure for random key selection using LFSR:

```
For (each 16-bit plain text block, PT)
{
unsigned int Random[16], key[16],
Reordered-key [16], LFSR16-seed;
unsigned char LFSR4-seed, Sequence-no;
LFSR 4-seed = Lower Nibble (LFSR 16 seed);
for (i = 0;i<16;i++)
{Random [i] = LFSR-4bit (LFSR-seed); }
for (i = 0;i<16;i++)
{
Sequence-no = Random [i];
Reordered-key [i] = key [Sequence-no];
}
      Encryption(PT);
}
```

The process of dividing the 256 bit key (KEYM) into four 64-bit keys(Ki) a shown in Fig. 3.

Similarly 16-bit sub keys can be obtained by sub-dividing each 64 bit key (Ki) into four 16-bit keys (ki,j). The order of these 16 sub keys are determined by the array ranom numbers in the array "Rand [16]" obtained from the output of the 4-bit LFSR. Based on the sample calcuations shown above as the first random number generated by 4-bit LFSR is 7, the sub key (7) i.e., Fig. 3. k 2,4 is taken as the first 16-bit sub key for the encryption process. The next number in the array is 3 therefore, the sub key (3) i.e., k 1,4 is taken as the second key. The sample output of the 4-bit LFSR is shown below.

Now these randomized keys are used while encrypting the given plain text as explained in the above.

Original key sequence

Subkey 0 K1,1	Subkey 1 K1,2	Subkey 2 K1,3	Subkey 3 K1,4
Subkey 4 K2,1	Subkey 5 K2,2	Subkey 6 K2,3	Subkey 7 K2,4
Subkey 8 K3,1	Subkey 9 K3,2	Subkey 10 K3,3	Subkey 11 K3,4
Subkey 12 K4,1	Subkey 13 K4,2	Subkey 14 K4,3	Subkey 15 K1,4

Randomized key sequence

Subkey 0 K2,4	Subkey 1 K1,4	Subkey 2 K1,2	Subkey 3 K3,1
Subkey 4 K2,1	Subkey 5 K1,3	Subkey 6 K3,2	Subkey 7 K4,1
Subkey 8 K2,4	Subkey 9 K2,4	Subkey 10 K2,4	Subkey 11 K2,4
Subkey 12 K4,2	Subkey 13 K4,3	Subkey 14 K4,4	Subkey 15 K1,1

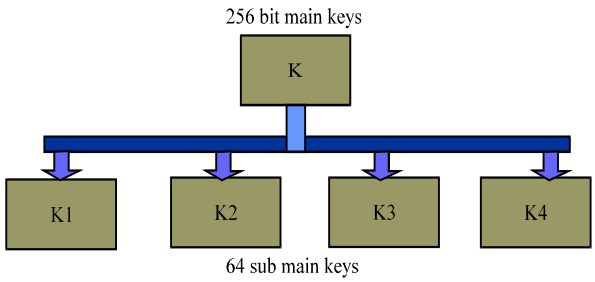

Fig. 3: 64-bit sub-key generation

In decryption also this modification has to be done to obtain the required plain text. This will increase the complexity and there by the security is raised to a great extend.

The above said algorithm was implemented in software using the Embedded 'C' language with the 8-bit microcontroller ATmega8 as the target device. The device features with Reduced Instruction Set Computers (RISC) architecture that supports throughput of about 1 MIPS/1 MHZ. The target device is well known in the application areas that demands low power consumption and low cost. The execution was tested with AVR GCC compiler on an open source Integrated Development Environment (IDE), AVR studio.

RESULTS

The simulation results obtained from the AVR studio on memory footprint and execution time for the original version of the hummingbird and the modified version with randomized sub-key selection using LFSR were taken for comparative analysis.

Table 2 and 3, respectively gives the results on FLASH Memory occupancy and execution time for code size and execution speed optimizations done by the compiler. All these values are taken for encryption or decryption of one block of data.

Sample I/P and O/P for basic algorithm:
Plain text:
HI (or) 'H' = 72; 'I' = 73
Obtained cipher text:
=.% (or)'.' = 132; '%' = 37

Sample I/P and O/P after randomizing key sequence:
Plain text:
HI (or) 'H' = 72; 'I' = 73
Initial key sequence:
THIS-IS-KEY-FOR-ENCRYPTION-PROCS
Randomized key sequence:
ONFOTIY-YP-PROCSR-S-ISENKE-ICRTH
Obtained cipher text:
JL (or) 'J' = 100; 'L'

Table 2: With size optimization

Algorithm	Flash memory size (bytes)	Execution time (m sec)
Humming bird		
Encryption	1896	215.289
Decryption	2004	214.806
Humming bird with RSS using LFSR		
Encryption	2456	216.837
Decryption	2392	215.222

Table 3: With time optimization

Algorithm	Flash memory size (bytes)	Execution time (msec)
Humming bird		
Encryption	4626	74.519
Decryption	5280	69.047
Humming bird with RSS using LFSR		
Encryption	6088	82.266
Decryption	5824	72.684

CONCLUSION

The present results show the 4-bit LFSR employed to randomize the sub-key usage improves the security level of the hummingbird algorithm by 2^4 times. The inference from the above graphs in Fig. 4-7 makes it clear that this hike in security can be achieved without significant raise in the execution time. Though the code size for the

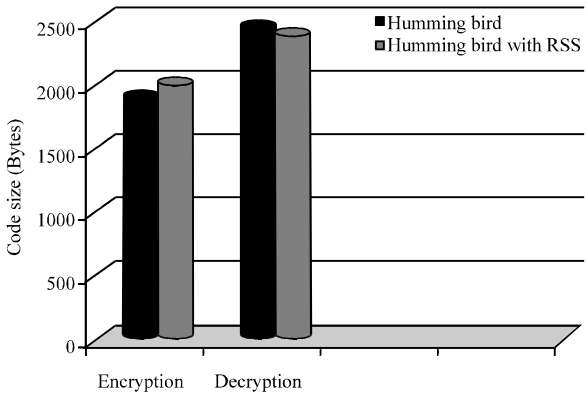

Fig. 4: Memory footprint comparison-code optimized

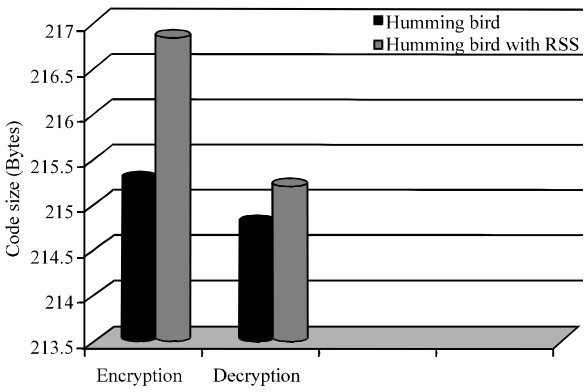

Fig. 5: Execution time comparison-code optimized

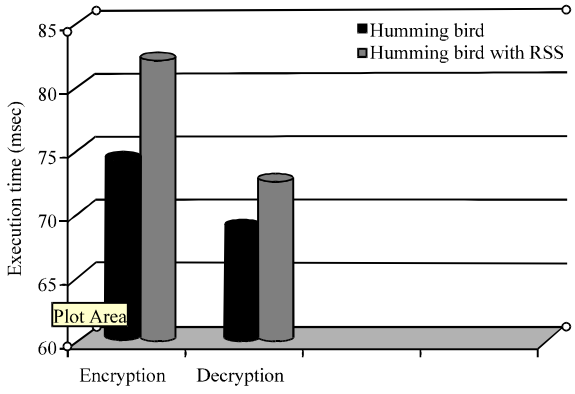

Fig. 6: Execution time comparison-time optimized

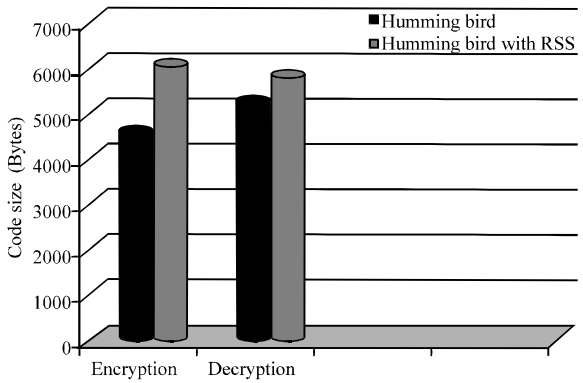

Fig. 7: Memory footprint comparison-time optimized

modified algorithm is slightly high, it does not make any negative impact on algorithm since the memory footprint is very less compared to the existing code memory area of ATmega8.finally it is clear that considerable improvement in execution time can be obtained by using compiler specific optimizations on AVR studio.

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.

Daemen, J. and V. Rijmen, 2002. The Design of Rijndael: AES-the Advanced Encryption Standard. Springer-Verlag, Berlin.

Diffie, W. and M. Hellman, 1976. New directions in cryptography. IEEE Trans. Inform. Theory, 22: 644-654.

Engels, D., X. Fan, G. Gong, H. Hu and E.M. Smith, 2010. Hummingbird: Ultra-lightweight cryptography for resource-constrained devices. Fin. Cryptogr. Data Security, LNCS, 6054: 3-18.

Gura, N., A. Patel, A. Wander, H. Eberle and C.S. Sheueling, 2004. Comparing elliptic curve cryptography and rsa on 8-bit cpus. Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems, LNCS., 3156, August 11-13, 2004, Springer, Berlin, Heidelberg, pp: 119-132.

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.

Leander, G., C. Paar, A. Poschmann and K. Schramm, 2007. New lightweight DES variants. Proc. Int. Workshop Fast Software Encrypt., 4593: 196-210.

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. Mahmod and M.R. Sulong, 2010. Key transformation approach for rijndael security. Inform. Technol. J., 9: 290-297.

Rabah, K., 2005a. Secure implementation of message digest, authentication and digital signature. Inform. Technol. J., 4: 204-221.

Rabah, K., 2005b. Theory and implementation of data encryption standard: A review. Inf. Technol. J., 4: 307-325.

Rajagopalan, S. and H.N. Upadhyay, 2011. Stego system on chip with LFSR based information hiding approach. Int. J. Comput. Appl., 18: 24-31.

Rajagopalan, S., R. Amirtharajan, H.N. Upadhyay and J.B.B. Rayappan, 2012a. Survey and analysis of hardware cryptographic and steganographic systems on FUGA. J. Applied Sci., 12: 201-210.

Rajagopalan, S., S. Janakiraman, H.N. Upadhyay and K. Thenmozhi, 2012b. Hide and Seek in Silicon-Performance Analysis of Quad Block Equisum Hardware Steganographic Systems. Procedia Eng., 30: 806-813.

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.

Roman, R., C. Alcaraz and J. Lopez, 2007. A survey of cryptographic primitives and implementations for hardware-constrained sensor network nodes. Mobile Networks Appl., 12: 231-244.

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., Wiley, India.

Zaidan, A.A., B.B. Zaidan, A.K. Al-Frajat and H.A. Jalab, 2010a. An overview: Theoretical and mathematical perspectives for advance encryption standard/ rijndael. J. Applied Sci., 10: 2161-2167.

Zaidan, B.B., A.A. Zaidan, A.K. Al-Frajat and H.A. Jalab, 2010b. On the differences between hiding information and cryptography techniques: An overview. J. Applied Sci., 10: 1650-1655.