Efficient Implementation of Grand Cru with TI C6x+ Processor

Azhar Ali Khan[1], Ghulam Murtaza[2]

[1] Sichuan University, Chengdu, China
[2] National University of Sciences and Technology, Islamabad, Pakistan
[1] azhar_iiee@yahoo.com, [2] azarmurtaza@hotmail.com

Abstract. Grand Cru, a candidate cipher algorithm of NESSIE project, is based on the strategy of multiple layered security and derived from AES-128. This algorithm was not selected for second phase evaluation of NESSIE project due to implementation and processing cost. In this paper we present relatively a fast implementation of the cipher using Texas Instrument's DSP C64x+.

Keywords: Grand Cru, Keyed Structure of AES-128, DSP Implementation of Grand Cru.

1 Introduction

There are many significant developments in cryptanalysis of Block Ciphers during past years and new techniques [2, 3, 4, 5, 6, 7, 8, 9] have been introduced. Many of them are effective enough to break full round versions of ciphers [3, 4, 5]. However these techniques are mostly successful on ciphers with fixed structure and components. For well designed key dependent or dynamic structured ciphers or ciphers having key dependent or dynamic components are mostly not mucheffected. Although there are some efforts for cryptanalysis of such kind of ciphers [12, 13, 14] but still these techniques are not mature enough to present serious vulnerability of the cipher. The vulnerability found in these type of ciphers are mainly due to poor design having some specific fixed observation in the cipher itself [14]. There is no reason to not believe that such a well designed cipher persist more security measures against modern cryptanalytic techniques.

Grand Cru [9] a candidate cipher algorithm of NESSIE [10] project, is successor of AES-128 [1] and is based on strategy of multiple layered security. This tactic is aimed at combining different security motivations and hence is a useful practice to improve security of widely used existing crypto algorithms against known modern cryptanalytic techniques. It also provides a noble way to use long keys in existing algorithms without redesigning key-schedule and/or encryption algorithms. This idea seems noble until and unless the proposed cipher is as efficient as the original one. Although the proposed algorithm was not selected for second phase of NESSIE project due to the cost of speed, no weakness was found in the algorithm.

TMS320DM648 belongs to Texas Instrument (TI) Advance Digital media processors. TheDM648 is based on the third-generation high-performance, advanced VelociTI™ very-long-instruction-word (VLIW) architecture [12]. This processor very useful for video/audio applications. DM648 contains CPU, memory and interrupts controllers. In addition, the DSP subsystem acts as the overall system controller, responsible for handling many system functions such as system-level initialization, configuration, user interface, user command execution, and connectivity functions.

Due to its subsystem, DM648 is an excellent choice for digital media applications [12].DSP's advanced VLIW CPU allows executing up to eight instructions per cycle. On these bases DM648 claim that this type of processor's performance is up to ten times the performance of typical DSPs.

DM648 contains comprehensive components including its mega module with memory (DSP subsystem), five video ports, Ethernet components, different I/O port and powerful DMA with DDR2 memory controller interface [15].

The complete Block diagram of DM648 is as shown in figure 1:

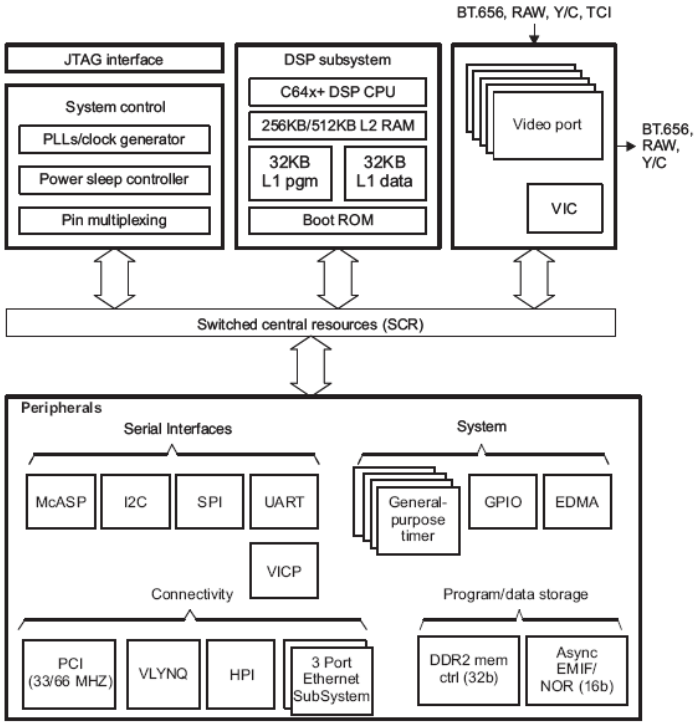

Figure 1: Block Diagram of DM648

The rest of the paper is organized as follows. A short description of Grand Cru algorithm is given in section 2. In Section 3, we describe the implementation

methodology of Grand Cru using TI DM648 processor. Section 4 briefly presents comparison of the implementation costs of AES-128 and Grand Cru. In section 5, we present the potential of designing multiple layered security based ciphers by the use of current state of the art hardware solutions.

2 Short overview of Grand Cru

In this section we present a short overview of Grand Cru [9]. We describe encryption algorithm and round keys generation procedures for the cipher.

2.1 Encryption Algorithm

Each round of encryption function of Grand Cru consists of the following sub-functions.

Initial/Outer Round Key Addition, ψ
A 128-bit sub key is added over modulo 2^8 (A Byte Wise Addition)

The transformation, v
The transformation $v: \{0,1\}^{128} \rightarrow \{0,1\}^{128}$ is defined as follows: Let the input of v be denoted by $x_{15} \dots x_0$ where each $x_i; i = 0,1, \dots 15$ is a byte then output $x_{15} \dots x_0$ is derived iteratively by:

1. for $i = 0\ to\ 15$ do
   $x_{i+1\ mod\ 16} = x_{i+1\ mod\ 16} \oplus S(x_i)$
2. for $i = 0\ to\ 15$ do
   for $j = 0\ to\ 15, j \neq i$ do
   $x_j = x_j \oplus S(x_i)$

Add Round Key, σ
An exclusive-or with the round key

Non Linear Layer, γ
Byte wise substitution of input using $8 - bit \times 8 - bit$ S-box

Keyed Byte Wise Rotation, β
Key-bits based rotation of output of S-box (a byte value)

Keyed Shift Row, π_R
A cyclic shift of the i^{th} row by j number of bytes to the left, where $i = 0 \dots 3$. Key determines the number of rotations of each row.

Keyed Shift Column, π_C
A byte of each column is permuted, column wise, which depends on the value of a key.

Mix Column Transformation, θ
An MDS matrix multiplication applied to each column. The final round does not include this transformation.

The transformation, $\mathbf{v^{-1}}$
The inverse transformation $v^{-1}$is given as

1. for $i = 15\ to\ 0$ do
   for $j = 0\ to\ 15, j \neq i$ do
   $x_j = x_j \oplus S(x_i)$
2. for $i = 15\ to\ 0$ do
   $x_{i+1\ mod\ 16} = x_{i+1\ mod\ 16} \oplus S(x_i)$

2.2 Round Keys Derivation

The key derivation of Grand Cru is different from AES key structure. The designer claims that only 128-bit current round key is required while the next round key will be derived on the fly from previous key.
The key expansion is based on the function $f_{rk}$and round keys are derived from user key by using following function.

$$K_i = f_{rk}(K^u, i)$$

Where K^u is a 128-bit user key, K_i is the round key for $i - th$ round.

The details of key derivation are as follows:

The user computes 128-bit key, K_0 using the following equation.

$$k_{33}k_{32}k_{31}k_{30}k_{23}\ldots k_{01}k_{00} = \begin{pmatrix} k_{03} & k_{02} & k_{01} & k_{00} \\ k_{13} & k_{12} & k_{11} & k_{10} \\ k_{23} & k_{22} & k_{21} & k_{20} \\ k_{33} & k_{32} & k_{31} & k_{30} \end{pmatrix} = (k_3|k_2|k_1|k_0)$$

where K_{ij} is a byte and K_i is a 4-byte value and $k_i; i = 0,1,2,3$ is the $i - th$ column. This user key is passed to the function f_{rk}, which applies column wise rotation as follows..

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \xrightarrow{Rotl} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_0 \end{pmatrix}$$

After the column rotation a substitution Γ will be done as

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \xrightarrow{\Gamma} \begin{pmatrix} S(x_0) \\ S(x_1) \\ S(x_2) \\ S(x_3) \end{pmatrix}$$

The next key is generated by passing the matrix through the following function.

$$K_i' = K_{i-4}' \oplus \Gamma\big(Rotl(K_{i-1}')\big) \oplus C_{i/4}\ if\ 4|i$$

$$K_i' = K_{i-4}' \oplus K_{i-1}'\ else,$$

Where $C_i$is the round constant. The 11 round keys are defined as

$$K_{10}, \dots K_1, K_0 = (K_{43}'|K_{42}'|K_{41}'|K_{40}'), \dots, (K_7'|K_6'|K_5'|K_4'), (K_3'|K_2'|K_1'|K_0')$$

The Grand Cru requires 4 different 128-bit master keys. These keys can be derived from a single 128-bit key by using some key derivative function. These four 128bit keys are then used in four round operations namely ψ , σ.π and β.

First 128-bit key is used for generating two 128-bit keys for ψ operation by using following equation.

$$K_0^3, K_1^3 = f_{rk}(K^3, 0), f_{rk}(K^3, 1)$$

Second128-bit key is used for derivation of 11 round keys by using the following function.

$$K_0^0, \dots, K_{10}^0 = f_{rk}(K^1, 0), \dots, f_{rk}(K^1, 10)$$

Third key generates 5 sub keys for each round and these keys are used in π. This operation consists of two (row and column) rotations. 5 sub-keys are generated as follows.

for $i = 0\ to\ 15$ do

1. Compute $k_{15} \dots k_0 = f_{rk}(K^1, i)$
2. Set counter $ctr = 0$
3. For $j = 0\ to\ 15$ do
   If $k_j < 24$ then $K_i^1[ctr] = k_j, ctr++$
4. If $ctr < 5$
   For $j = ctr\ to\ 5$ do $K_i^1[ctr] = k_{j-ctr}\ mod\ 16$

The last 128-bit key is used in β transformation which requires sixteen 3-bit keys for 10 rounds as described under

for $i = 0\ to\ 9$ do

1. Compute $k_{15} \dots k_0 = f_{rk}(K^2, i)$
2. $K_i^2 = LSB_3(k_{15}) \dots LSB_3(k_0)$

3 Implementation of Grand Cru

T1 C6x+ processor is one of the best processors for implementation of different algorithms. The round keys derivation and encryption algorithm implementation techniques are presented in the following sub sections.

3.1 Round Keys Derivation Implementation Technique

The implementation of round key function in TI DM648 processor requires ROTL, ADD4, XOR, LDW and conditional instructions. Look-up table is used for substitution. The core of round key derivation of Grand Cru consists of a conditional function and XOR operation. The conditional function checks if the round number is exactly divisible by 4 then the intermediate function is executed otherwise XOR operation is required. The intermediate function comprises a rotation of 32-bit value (column) with substitution and two XOR operations. The 32-bit registers of the processor helps in implementing the above function.

As substitution of Grand Cru is Byte wise, so implementation requires a splitting process from 32-bit into 8-bit values. The implementation of f_r function takes the advantage of parallelism of DM648. The load and store operations are done in parallel. So, the f_r function takes 80 cycle/called. The f_r is used in key generation for functions ψ , π , β and σ. The functions σ and ψ, need to just call f_r function, where as the functions π and β require other mathematical and logical operations.

Theψ‘s keys generation call f_r only once which implies that the process consumes only 98 cycles. Variables adjustment and function call, consume 18 cycles. The σ calls f_r 10 times and use 494 cycles. The π ‘s key generation function calls the function f_r 9 times, which requires 450 cycles. It also has ‘For’ loops with conditional instructions. These extra operations require 4898 cycles. Hence a complete generation of π requires 5348 cycles. The β transformation requires sixteen, 3-bit keys per round. It calls the function f_r, 9 times and trims the output into 3 bits. The required number of cycles for key generation of function β is 1071.

The number of cycles required for Grand Cru's key generation is shown in Table 1.

Sr#	Operation(key generation)	Cycles
1	ψ	98
2	σ	494
3	π	5348
4	β	1071
Total		**7011**

Table 1: Number of Cycles for Key Generation Algorithm

3.1 Encryption Algorithm Implementation Technique

Initial/Outer Round Key Addition, ψ

This transformation is byte-wise modulo addition and Grand Cru calls this addition two times. These two transformations can easily be implemented with DM648 instruction ADD4. Each transformation with keys loading takes 22 cycles.

The transformations v and v^{-1}

From conceptual point of view, this transformation is very simple but this is the most expensive one from implementation aspect. Each transformation includes two intermediate functions and each function has two 'For' loops. The first loop has 15 substitution and XOR operations. The second loop is nested one and has a total 256 substitution and XOR operations with conditional instruction.

After the implementation of $\boldsymbol{v}$ functions, we found that the function $\boldsymbol{v}$ takes 4352 cycles and the function $\boldsymbol{v^{-1}}$ requires 4414 cycles.

Add Round Key, σ

This function of Grand Cru applies a XOR function to the input data (128-bit) and the corresponding round key. DM648 architecture gives LDDW (64-bit load) instruction to load 128-bit key and use only two LDDW. This architecture also reduces multiple XOR operations and need only 4 XOR instructions for 128bit value. The parallel unit of DM648 makes it possible that these XOR operations operate in parallel with other instructions. Hence these operations logically consume zero cycle.

Non Linear Layer, γ

The simple and efficient method for this function is to use Look-up table which requires addition and load instructions. The output of Add Round Key function is

added with look-up table's (32-bit base address) and gets exact location of required data. After the address obtain, its value is loaded.

Keyed Byte-wise rotations, β

The function **β is** based on K^2. The rotation is applied on single byte so it does not allow simple rotation instruction, ROTL. The shift operations are useful for this purpose. The number of rotations in left shift, SHL is based on key-bit value. The SHL contains the shifted bits of rotated byte while applying XOR it with right shifted (SHR) byte and gets the desire output.

Instead of developing two separate modules of γ and β, the efficient single module is developed by using parallel instructions of C6x+. This single module consumes 128 cycles/round. Another advantage of this module is that the output of this module is 32-bit that is useful for next step.

Keyed Shift Row, π_R

In Grand Cru, there are 2 keyed rotations which rotate a byte value within a 32-bit value. The 1st keyed rotation is applied row-wise after default shift row operation as in AES. This rotation takes place by the use of 3rd key of Grand Cru. Grand Cru represents a table, 't-table' to ensure a unique rotation in all blocks of 32-bit values in single round.

For pi rotation, a byte of key splits into pairs of 2-bit values which takes 7 cycles. The VLIW's instruction ROTL takes 2 cycles for each 32-bit values and splitting applies bit-wise. Thus, value of shift byte converts into bits and then it uses with ROTL. The rotation takes 2 cycles for each column but generating keyed rotation value take 7 Cycles/Round. The complete Keyed shift Row takes 16 cycles/round.

Keyed Shift Column, π_C

The 2nd keyed permutation in Grand Cru algorithm is keyed column Shift. The byte store in keyed Shift Row is row wise and ROTL instruction in column shifting requires conversion of rows into columns. After conversion, this shift needs a single ROTL instruction. With all the complication, it takes 71 cycle/round.

Mix Column Transformation, θ

Mix column in Grand Cru is same as in AES. VLIW offers a valuable instruction Galois filed multiplication (GMPY). Its modified instruction, GMPY4 applies on byte. Before applying GMPY4, irreducible polynomial in GFPGFR needs to set. The output of GMPY4 gives 32-bit value and for XOR sum of these 4 bytes requires

splitting of 32-bit into 8-bit. This Splitting can done by using C6x+ instructions AND, SHL/SHR, and XOR. GMPY4 instruction requires 3 delay slots which is meaningless by whole matrix multiplication. Due to splitting, one element of mix column will generate in 10 cycles and sixteen elements need 160 cycles. Here parallel unit gives full support to reduce cumulative mix column's cycles and it consume 109 cycles per round.

Sr.#	Operation(encryption)	Cycles/Round
1	ψ	22
2	V	4352
3	v-1	4414
4	σ	0
5	γ and β	128
6	π_R	16
7	π_c	71
8	θ	109
Total	**128-bit Encryption**	**11946**

Table 2: Number of Cycles for Encryption Algorithm

Some cycles of individual operations has been reduced by using parallel operation with other modules.

4 Comparison

The designer of Grand Cru implemented the algorithm on 8-bit Motorola microcontroller and used ANIS C on Pentium and Xeon. Advance digital signal processors reduce implementation cycles and drastically increase the throughput. The following table shows the comparison of results of processing cost of grand cru over different platforms.

	Encryption/Cycle	**Key setup/Cycle**
8-bit Motorola	60000	300000
Pentium	45000	200000
Xeon	65000	300000
DM648	11946	7011

Table 3: Cost Comparison of Grand Cru over Different Platforms

The comparison of implementation of Grand Cru with AES-128 in DM648 is as shown in the following table.

	AES-128	Grand Cru	Grand Cru excluding v and v^{-1}
Key setup/Cycle	651	7011	7011
Encryption/Cycle	2197	11946	3180

Table 4: Cost Comparison of AES-128, Grand Cru and Grand Cru excluding v and v^{-1} functions using DM648

Table 4 shows that the processing speeds are 40Mbps, 7Mbps and 27.6 Mbps @720 MHz for AES-128, Grand Cru and Grand Cru excluding v and v^{-1} functions.

5 Conclusion

We have presented a reasonably efficient implementation of Grand Cru [9] using current Texas instruments T1 DM648 processor. We conclude that with the advancement in hardware, we are able to implement ciphers with proceed structures in efficient manner. One of our conclusions is that choices of sub-function in cipher are mainly cause of decreased efficiency rather than multiple layered structures itself. For example, by removing v and v^{-1} function we have distinctly increased in efficiency of Gran Cru. Further our effort smashes the restriction in designing of multiple layered security based ciphers that are due to cost issues and one can design such ciphers to enhance security against rapidly increasing cryptanalytic techniques of conventional block ciphers. One also can obtain remarkably better results from our implementation by using techniques in efficient implementations of AES.

References

1. FIPS-197: Advanced Encryption Standard, November 2001, available at http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
2. Biham, E., Dunkelman, O., Keller, N.: Related-key impossible differential attacks on AES-192, In CT-RSA'06, LNCS, vol. 3860, pp. 2--31, Springer Verlag (2006).
3. Biryukov, A., Khovratovich, D., Nikolic, I.: Distinguisher and related-key attack on the full AES-256. In CRYPTO'09, LNCS. Springer Verlag (2009).
4. Biryukov, A., Khovratovich, D.: Related-key Cryptanalysis of the Full AES-192 and AES-256, IACR ePrint report 2009/317, 2009. Available online at http://eprint.iacr.org/2009/317.
5. Biryukov, A., Dunkelman, O., Keller, N., Khovratovich, D., Shamir, A.: Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds. Cryptology ePrint Archive, Report 2009/374, 2009. Available online at http://eprint.iacr.org/2009/374.
6. Kim, J., Hong, S., Preneel, B.: Related-key rectangle attacks on reduced AES-192 and AES-256. In FSE 2007, LNCS, vol. 4593, pp. 225-241, Springer Verlag (2007).
7. Yong Zhuang, W., YuPu, H.: New Related-Key rectangle attacks on reduced AES-192 and AES-256, Science in China Press, vol.52, n.4, pp. 617-626, Springer Verlag (2009).

8. Biham, E., Keller, N.: Cryptanalysis of Reduced Variants of Rijndael, Available at http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html
9. Borst, J.: The bolck cipher: Grand Cru, Available at https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions/grandcru.zip.
10. NESSIE (New European Schemes for Signatures, Integrity and Encryption) was a European research project funded from 2000–2003 to identify secure cryptographic primitives. https://www.cosic.esat.kuleuven.be/nessie/
11. Rimoldi, A.: PhD Thesis: On algebraic and statistical properties of AES-like ciphers. PhD thesis, University of Trento (2009) Available at http://eprints-phd.biblio.unitn.it/151/
12. Gilbert, H., Chauvaud, P.: A Chosen Plaintext Attack of the 16-round Khufu Cryptosystem, In CRYPTO '94, LNCS, vol. 839, pp. 359-368, Springer Verlag (1994).
13. Vaudenay, S.: On the Weak Keys of Blowfish, In FSE '96, LNCS, vol. 1039, pp. 27-32, Springer Verlag (1996).
14. Abdelraheem, M. A., Leander, G., Zenner, E.: Differential cryptanalysis of round-reduced PRINTcipher: Computing roots of permutations. In Antoine Joux, editor, Proc. FSE 2011, LNCS, vol. 6733, Springer Verlag (2011).
15. TMS320Digital Media processor 648, Literature Number SPRS372F.
16. TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide, Literature Number: SPRU732H October 2008.