2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)

Macros for Coding Information Encryption & Decryption in Trusted Platform Module

V. Ruchkin, G. Soldatov, V. Fulin
CS and Engineering Dept.
Ryazan State University
Ryazan, Russian Federation
v.ruchkin @rsu.edu.ru

B. Kostrov, E. Ruchkina
Computer Dept.
Ryazan State Radio-Engineering University
Ryazan, Russian Federation
kostrov.b.v@evm.rsreu.ru

***Abstract*—This article describes the process assembling of Encryption & Decryption In Trusted Platform Module of encoding information in NM640X ®. Encoding of information, carried out in assembly language according to Gost 28147-89. It is a realisation of standard GOST 28147-89- Russian state symmetric key block cipher. GOST 28147-89 has 64-bit to access the kernel, trust, and allocated memory in the BlockSize and 256-bit KeySize**

Keywords— encryption; decryption; trusted platform module; Decoding Assembly Language; Macros; OS Windows Security; DES; GOST; AES.

I. INTRODUCTION

In any computing device, the process of converting the original data M (message) in the form of "original text" into encoded or encrypted C (cipher text) data occurs repeatedly. This process is commonly referred to as encryption or encryption using the encryption function E (encryption). Mathematically, this transformation is represented by a dependence that describes all actions on the initial information [1]:

$$C = Ek_1(M)$$

The reverse process of converting encrypted C (cipher text) or M' – information obtained as a result of decryption into open data M (message) by means of the decryption function D (decryption) is called decryption, respectively, and is mathematically written as a dependence:

$$M' = Dk_2(C),$$

where k_1 (key) is a parameter of the function E, called the encryption key and k_2 is the key - a parameter of the function D.

The encryption and decryption technologies are based on Feistel networks [2], which allow developing an algorithm that meets all the requirements for symmetric encryption algorithms with sufficient simplicity and compactness. Moreover, the Feistel network is reversible even if the generating function F is not. For decryption, the same algorithm is used, but an encrypted message is sent to the input and the keys are used in reverse order. The disadvantages include the dependence of the encryption security on the number of bits of the original information and keys, which slows down the process of encryption and decryption. However, Feistel networks underlie the whole set of coding algorithms that are constantly being improved, for example: DES, GOST, to some extent AES.

The GOST 28147 algorithm is a classical Feistel network with a mathematical formula [1]:

$$L_i = R_i$$
$$R_i = L_i + F(R_i - 1, K_i)$$

and is a domestic standard for symmetric encryption algorithms with a block length of 64 bits, a key length of 256 and the number of rounds is 32.

GOST 28147-89 is the Russian standard of symmetric encryption introduced in 1990. Full name: "GOST 28147-89 Information Processing system". This is an algorithm of block cipher, and in case of using the method of encryption with XOR cipher can perform functions of multi-threaded algorithm of cipher.

Formexample, you can use this method in Technology Virtualization Based Security (VBS), two important security enhancements can be implemented based on the module's Multi-programmable On Chip Processors (MPoC) 77.07 and MC 127.05 [2-3]: A new trust boundary between key components of the OS system Windows 10 and the secure virtual runtime in which they run [4].

Technology VBS Provides a secure runtime environment because the architecture of this environment does not allow processes running in the environment Windows have full system privileges environment VBS [4]. In addition, the environment VBS Uses TPM 2.0 to protect data stored on disk. Similarly, a user who has access to a physical disk cannot access the data unencrypted.

Trusted Platform Module (TPM) is a fault-tolerant module that enhances the security and privacy of computing platforms. TPM Integrated into a reliable computing platform (PC, tablet or Phone) as one of its components. The computing platform is designed to work with TPM that provides security and privacy that cannot be achieved by using the software alone. Correct implementation TPM as part of a trusted computing platform, it generates a hardware trust kernel that indicates that the hardware behavior is trusted. For example, a key created in

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

TPM with a property that prohibits the export of this key from TPM means that the key will never be able to leave the TPM. Tight integration TPM with the platform, it enhances the transparency of the information download process and supports device state validation scenarios by compiling a reliable software report that is used to run the platform [6].

II. TPM Functions

Implementing Functionality TPM you must create the following components:

- Key Management: creating, saving, and allowing the use of keys: DES, GOST, AES in a particular way in different attack situations.
- Protection and reporting on integrity indicators. The software used to load the platform can be TPM and used to establish trust relationships in software running on the platform.
- Proof of TPM capability as a key role in protecting the privacy and security required to highlight malware, masks in TPM.
- Biometric protection.

Microsoft Combined these and other benefits TPM With OS Windows 10 and other hardware security technologies compared to other existing operating systems, thus providing significant security and privacy benefits.

In OS Windows 10 Technology Technologies TPM is used to protect volume encryption keys BitLocker, virtual smart cards, certificates, and many other keys that you can use to create TPM. Windows 10 also uses TPM to reliably record and protect the integrity of your favorite hardware and boot components Windows. In this scenario, the measured load measures each component (from Teh firmware to disks) and then saves these indicators in the module TPM on the PC. From here you can remotely check the performance log, i.e. check the boot status of the PC running the OS Windows 10 using a separate system [7].

Usually TPM It is installed on the equipment or motherboard as a separate module. In this work it is suggested to use the microprocessor module MV 77.07 and MC 127.05 on the basis of the digital signal core of the processor NeuroMatrix with 32-bit scalar RISC and 64-bit vector processors with architecture VLIW and SIMD. As a result of the use of such module any computer system will have increased security and confidentiality by means of OS Windows 10, which does not distinguish between separate modules and modules in the embedded software. Therefore, any component Windows where you can use TPM can use any module implementation.

III. Result. Management of Keys of Encryption and Decryption

A. MultyProgrammable on Chip

As its main processor, the MultyProgrammable on Chip (MPoC) uses a SBIS K1879KhB8Ya microchip, developed by ZAO NTTs Module. The SBIS K1879KhB8Ya performs decoding tasks in transport and software flows, video decoding [8], including HD video acc. to MPEG4- 10/H, 264/AVC standards, HP/L4.1, MPEG2, MP/HL, SMPTE 421M/VC-1 AP/L3, general management of systems and support of the user interface.264/AVC standards, HP/L4.1, MPEG2, MP/HL, SMPTE 421M/VC-1 AP/L3, general management of systems and support of the user interface (Fig. 1.).

Fig. 1. MC 127.05 modules with the 1879VM8Y chip

B. Encryption

Encryption according to GOST 28147-89 consists in repeated repetition of the basic step of the algorithm Crypto-formation, which is the operator of the definition of transformation of the 64-bit block of data. An additional parameter of this operator is the 32-bit block, which is used as a key element.

Consider data encryption for N = HYDROGEN and key K = ABCDEFGH KLMN are encoded using a table ASCII [7]:

TABLE I. DATA

N	H	Y	D	R	O	G	E	N
	H	Y	D	R	O	G	E	N
N	8	9	4	2	F	7	5	E
	8	9	4	2	F	7	5	E

TABLE II. KEYS

K	A	B	C	D	E	F	G	H	K	L	M	N
	A	B	C	D	E	F						
	1	2	3	4	5	6	7	8	B	C	D	E
	1	2	3	4	5	6						

Here is the code of Assembler program for encryption GOST 28147-89:

```
Global __main: label;
Data ". DataSeg
    N: Long = 0485944524 inf47454Ehl; Original word
    N_Sh: Long; Encrypted word
    K0: Long = 041424344HL; Key
    K1: Long = 045464748HL;
```

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

```
K2: Long = 04b4c4d4ehl; K: Long = 041424344454647485556 5758HL; // All key
Massh: Long = 0Fff00000Hl; 11-Bit Selection mask
Mask0: Long = 00000000FHL; // Masks for the selection of interchangeable tetades
Mask1: Long = 0000000F0HL;
Mask2: Long = 000000F00HL;
MASK3: Long = 00000F000HL;
MASK4: Long = 0000F0000HL;
MASK5: Long = 000F00000HL;
MASK6: Long = 00F000000HL;
Mask7: Long = 0F0000000HL;
Global A: Long [8] = (0D85A72B51F36E0C9HL, 0478E0BF12DC5A936HL, 0ABE69D387401F52CHL, 0FD87BEC32A946150hl, 06EC5A30BF3472918hl, 0B46F1D29E7C0A583hl, 08F96C73502ED4BA1hl, 0E9FD8ABC41567302HL);
Replacement table
Table: Word [128] = (13, 8, 5, 10, 7, 2, 11, 4, 1, 15, 3, 6, 14, 0, 12, 9,
        4, 7, 8, 14, 0, 11, 15, 1, 2, 13, 12, 5, 10, 9, 3, 6,
        10, 11, 14, 6, 9, 13, 3, 8, 7, 4, 0, 1, 15, 5, 2, 12,
        15, 13, 8, 7, 11, 14, 12, 3, 2, 10, 9, 4, 6, 1, 5, 0,
        6, 14, 12, 5, 10, 3, 0, 11, 15, 3, 4, 7, 2, 9, 1, 8,
        11, 4, 6, 15, 1, 13, 2, 9, 14, 7, 12, 0, 10, 5, 8, 3,
        8, 15, 9, 6, 12, 7, 3, 5, 0, 2, 14, 13, 4, 11, 10, 1,
        14, 9, 15, 13, 8, 10, 11, 12, 4, 1, 5, 6, 7, 3, 0, 2);
The end. " DataSeg ";
Macro block (MAS, num, SD)//Blocker macro replacement
    AR2 = Mas;
    GR5 = [AR2];
    GR6 = Gr5 and GR4;
    G7 = Table[Num]; // Item Replacement Address
    GR6 = GR6 > > SD;
    GR7 = GR7 + GR6;
    GR0 = [GR7];
    GR0 = GR0 < a1/&gt;
    GR1 = GR1 XOR GR0; Result
End Block;
Macro Step(Key)//Macro of one step of the algorithm//addition to the module 2 with the first key of the senior half
    Ar1 = theKey; Getting the key
    G3 = [Ar1]; // K
    GR4 = Gr2 XOR gr3; Step One
    G1 = 0;
    Block replacement;
    Block(Mask7, 0, 28);
    Block (MASK6, 16, 24);
    Block (MASK5, 32, 20);
    Block (MASK4, 48, 16);
    Block (MASK3, 64, 12);
    Block (Mask2, 80, 8);
    Block (Mask1, 96, 4);
    Block(Mask0, 112, 0);
//the G1 saved replaced by 32-bit//value, further offset by 11 bits
    AR7 = Massh;
    GR5 = [AR7];
    GR5 = GR5 and Gr1;
    GR7 = Gr5 > 21;
    GR5 = GR1 < A1/&GT;
    GR1 = Gr5 XOR Gr7;
    G6 = Ar6/ N2
G0 = G6 Xor G1;// redistribution of a cyclic repetition of the step
    GR2 = GR6;
    AR6 = Gr0; // N1-GR2, N2-AR6
// The final step;
Get started. " Tekstaaa "
<__Main>//Split the original word into 32-bit elements
    Ar0 = N;
    AR6 = [Ar0 + +];//n2
    GR2 = [AR0];//n1
    Step(K0);
    Step(K1);
    Step(K2); //Then write a word in memory
    Ar0 = N_sh;
    [Ar0 + +] = AR6;
    [AR0] = GR2;
    Return;
The end ".Tecstaa";
```

In the figure 3 Represents the contents of registers after the program runs. The color is marked by registers containing the senior (N1) and Junior (N2) Part of the encoded word. N1 is stored in the register G2 N2 – In the Register Ar6. Drawing 4 The encrypted word in memory is presented.

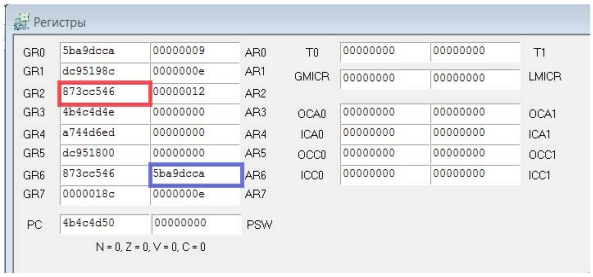

Fig. 2. The contents of the registers after the encryption program.

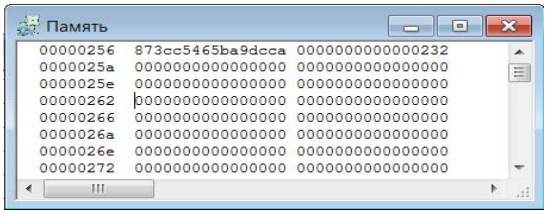

Fig. 3. Encrypted word in memory.

C. *Decrypt*

Here is the code of the Assembler program to implement the decryption of GOST 28147-89:

```
Global __main: label;
Data ". DataSeg
N: Long = 0873cc5465ba9dccahl;//original Word
N_desh: Long; Deciphering the word
 K0: Long = 041424344HL; Key
K1: Long = 045464748HL;
K2: Long = 04b4c4d4ehl;
K: Long = 041424344454647485556 5758HL; All key
```

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

```
Massh: Long = $ sFffhl; 11-Bit Selection mask
Mask0: Long = 00000000FHL; Masks for the selection of interchangeable tetades
Mask1: Long = 0000000F0HL;
Mask2: Long = 000000F00HL;
MASK3: Long = 00000F000HL;
MASK4: Long = 0000F0000HL;
MASK5: Long = 000F00000HL;
MASK6: Long = 00F000000HL;
Mask7: Long = 0F0000000HL;
Global A: Long [8] = (0D85A72B51F36E0C9HL, 0478E0BF12DC5A936HL, 0ABE69D387401F52Chl, 0FD87BEC32A946150hl, 06EC5A30BF3472918hl, 0B46F1D29E7C0A583hl, 08F96C73502ED4BA1hl, 0E9FD8ABC41567302HL);
Replacement table
Table: Word [128] = (13, 8, 5, 10, 7, 2, 11, 4, 1, 15, 3, 6, 14, 0, 12, 9,
        4, 7, 8, 14, 0, 11, 15, 1, 2, 13, 12, 5, 10, 9, 3, 6,
        10, 11, 14, 6, 9, 13, 3, 8, 7, 4, 0, 1, 15, 5, 2, 12,
        15, 13, 8, 7, 11, 14, 12, 3, 2, 10, 9, 4, 6, 1, 5, 0,
        6, 14, 12, 5, 10, 3, 0, 11, 15, 3, 4, 7, 2, 9, 1, 8,
        11, 4, 6, 15, 1, 13, 2, 9, 14, 7, 12, 0, 10, 5, 8, 3,
        8, 15, 9, 6, 12, 7, 3, 5, 0, 2, 14, 13, 4, 11, 10, 1,
        14, 9, 15, 13, 8, 10, 11, 12, 4, 1, 5, 6, 7, 3, 0, 2);
The end. " DataSeg ";
Block Replacement macro
Macro block (MAS, num, SD)
    Own vhillebegin: label;
    Own Vhile_end:label;
    G2 = [Ar1];
    G0 = 0;
    Ar2 = theMas; // Get Mask Address
    G5 = [Ar2]; // Get Mask Value
    G6 = G5 И G1; // scroll to the desired byte
G7 = Table[Num];//Retrieving the address of a table row
    G6 = G6 > >Sd; // Move byte to end of Word<A_Start>
G5 = [G7]; // Retrieving the value of the TABLE(0 row element)
G2 = the G6 Xor G5; //Checking for equality with a dedicated byte
If = 0 Goto A_The end;// If equal, the transition by the label
    G0 = G0 + 1; // Index increment
    [Ar1] = G0 // Write new value to variable Ind
        GR7 = GR7 + 1;  Go into the begin;
        GR0 = GR0 < a1//&gt;
    GR3 = GR3 XOR GR0;
End Block;
// Basic Step Macro algorithm
Macro Step(Key)
Step 1- N1 Xor N2
    Ar5 = G3; //copy for further use
G3 = the G3 Xor G2;//2 Step-cyclic Shift right to 11 bit
    AR7 = Massh;
    GR5 = [AR7];
    GR5 = GR5 and GR3;
    GR7 = GR5 < A1/&GT;
    GR5 = GR3 > 11;
    GR1 = Gr5 XOR Gr7;
    G3 = 0; //  replacing the pitch-reverse on the table
    Block (Mask7, 0, 28);
    Block (MASK6, 16, 24);
    Block (MASK5, 32, 20);
    Block (MASK4, 48, 16);
    Block (MASK3, 64, 12);
    Block (Mask2, 80, 8);
    Block (Mask1, 96, 4);
Block (Mask0, 112, 0);//in GR3 4Step-Addition with key
    Ar1 = the Key;
    GR0 = [AR1];
GR3 = GR3 XOR GR0; // Step 5-Redistribution, In G3 n1
    In G2 n2
    Ar0 = N;
    AR0 = AR0 + 1;
    GR2 = 5;
The end Step;
Start ".Tecstaa"
<__Main>//Split word in 32-bit elements
    Ar0 = N;
    AR6 = [Ar0 + +];//n2
    GR3 = [AR0];//n1
    GR2 = AR6;/n2
    Step (K2);
    Step (K1);
    Step (K0);
Ar0 = N_desh;
    [Ar0 + +] = GR2;
    [AR0] = GR3;
    Return
The end ".Tecstaa";
```

In the figure 5 Represents the contents of registers after the program runs. The color is marked by registers containing the senior (N1) and Junior (N2) Part of the encoded word. N1 is stored in the register G3 N2 – In the Register G2. Figure 6 The decoded word is represented in memory.

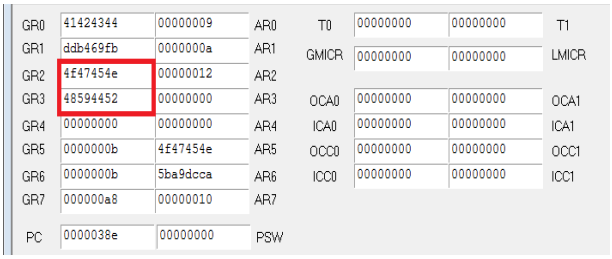

Fig. 4. Contents of registers after execution of decryption program.

```
000003d8  485944524f47454e 00000000000003b4
000003dc  0000000000000000 0000000000000000
000003e0  0000000000000000 0000000000000000
000003e4  0000000000000000 0000000000000000
000003e8  0000000000000000 0000000000000000
000003ec  0000000000000000 0000000000000000
000003f0  0000000000000000 0000000000000000
000003f4  0000000000000000 0000000000000000
```

Fig. 5. Deciphering words in memory

IV. Conclusion

The advantages of the standard are the futility of the brute-force attack, the efficiency of implementation and, accordingly,

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

high performance on modern computers, the presence of protection against the imposition of false data and the same encryption cycle in all four algorithms of the standard.

To demonstrate the management of keys, protect and report integrity, the virtual security environment, the cloud service to certify the health of the computing platform, was reviewed and Implemented biometric protection in practice one of the most common nowadays methods of encoding and decoding of information according to the algorithm of GOST 28147-89.

ACKNOWLEDGMENT

The paper was prepared as a result of the research conducted under state scientific grant № 2.9519.2017/8.9 "Technologies of parallelism of handling data in neuro computing ware and systems".

REFERENCES

[1] Horst Feistel, "A Survey of Problems in Authenticated Communication and Control", MIT Lincoln Laboratory, 1958.

[2] A. Chernikov, V. Chernikov, P. Vixne, A. Shelukhin, "High-Performance NMC4 Vector Processor Core For Fixed And Floating Point Calculations", Proceeding of 6th Moscow Supercomputing Forum 2015, pp. 13-14.

[3] A. Chernikov, V. Chernikov, P. Vixne, A. Shelukhin, "New Core Of Signal Processor Core NMC4 Of Set Neuro Matrix", Proceding of 6th Moscow Supercomputing Forum 2015, pp. 12-13.

[4] V.N. Ruchkin, V.A. Fulin, V.A. Romanchuk, "Personal Trusted Platform Module for the Multi-Core System of 5G Security and Privacy", Proceedings of the 13th International Conference ELEKTRO 2020, technically co-sponsored by Region IEEE 8, Czechoslovakia and Italian Section of IEEE on May 25th – 28th, 2020.

[5] A.N. Kolesenkov, B.V. Kostrov, V.N. Ruchkin, "The Neural Networks of the Monitoring of Anthropogenic Emergencies on the Base of the Earth Remote Sensing", Tula State University Review, Teachnical Sciences, Tula State University Press, 2014, Vol. 5, pp. 220-225. (In Russian)

[6] V.N. Ruchkin, B.V. Kostrov, A.N. Kolesenkov, E.V. Ruchkina, "Anthropogenic Situation Express Monitoring on the Base of Fuzzy Neural Networks", Proceedings of the 3nd Mediterranean Conference.

[7] A.N. Kolesenkov, B.V. Kostrov, V.N. Ruchkin, "The Neural Networks of the Monitoring of Anthropogenic Emergencies on the Base of the Earth Remote Sensing", Tula State University Review, Teachnical Sciences, Tula State University Press, 2014, Vol. 5, pp. 220-225. (In Russian)

978-1-7281-6951-4/20/$31.00 © 2020 IEEE