Automation System for Configuration of Cryptographic Data Protection Unit Model

Ivan Zholubak[0000-0001-8871-7222], Mohamed Rahma[0000-0002-8377-1833] and Valeriy Hlukhov[0000-0002-0542-7447]

Lviv Polytechnic National University, St. Bandera Str. 12, Lviv, 79013, Ukraine
IvanZholubak7@ukr.net

Abstract. Elliptic curves are mathematical basis for digital signature processing. In this case, the processing of the elliptic curve points is based on the operations in the Galois fields *GF(p^m).* Comparison of multipliers' hardware costs for Galois fields with different characteristics *p* is carried out in the work. The multipliers are intended for use as part of the cryptographic data protection system that is implemented on the FPGA. VHDL-descriptions of multipliers (cores) were created with the help of the developed core generator. It was found that hardware multipliers that process elements of the fields with characteristics 2, 3 and 7 *)* and with approximately equal order and the representation of these elements in a polynomial basis have a lower hardware complexity than multipliers for Galois fields with other characteristics. Software complexity of multipliers for Galois fields *GF(p^m)* with approximately equal order and the representation of these elements in a polynomial basis is also investigated. It was found that software multipliers that process elements of fields with characteristics 3, 5 and 7 have a higher time complexity (software complexity) than multipliers for Galois fields with other characteristics.

Keywords: Galois field GF(d^m), multiplier, modified Guild cell, LUT, core generator.

1 Introduction

In the implementation of algorithms for performing arithmetic operations in finite fields [9] or Galois fields GF(p^m) a large number of arithmetic and logical operations must be performed. The implementation of complex computational algorithms at the level of logic elements is a common practice [6].

Currently, in the practice of cryptographic data protection, logical fields GF(2^m) and simple fields GF(p) are used. Fields with the characteristic p > 2 - GF(p^m) are not used extensively. In this paper, hardware and software complexity of multipliers for Galois field elements GF(p^m) with different field characteristics *p* but with approximately the same order is compared.

In carrying out this work, on the basis of multiplier model proposed in [1 and 5] its implementation was carried out and the results of the hardware complexity estimation

(previously obtained theoretically) was obtained It was shown that the hardware complexity will be the smallest for the fields with the characteristic 2.

In [1, 2] a theoretical comparison of the hardware complexity of the multipliers of Galois fields GF(p^m) with different characteristics p was made. It was seen that multipliers for fields with characteristic 3 and 7 implemented on modern FPGAs that have logical blocks with 6 inputs and 1 output have the least hardware complexity. In [1] comparisons of Galois multipliers based on Modified Guild Cells (MGC) were considered. The MGC was considered to be an black box (read only memory, ROM) or a set of multiplier and adder [1]. In [2] estimation of hardware costs of Galois field multipliers in case when the MGC is consist of logical elements was made. The Galois field multiplier time complexity was also performed, advantage of Galois fields with field characteristics greater than 2 was shown and a comparison of the structural complexity [3] of the Galois fields GF(p^m) multipliers for different field characteristic p was made.

The most common methods of hacking computer systems are software brute force method, that is, a simple override of keys that can be implemented on supercomputers or quantum computers. Well known hacking methods were taken into account in determining the vulnerability of cryptographic data protection systems that use Galois fields with different field characteristics to hacking. In [5] a comparison of software implementations of the Galois multipliers was made. Software realizations of the multiplier for Galois field elements GF(2^m) are well known.

A multiplier core generator was developed to generate a multiplier for Galois fields with different characteristics and field orders. [4] describes methods for constructing of core generators.

The purpose of this work is to compare FPGA hardware and software costs of multipliers for Galois field with different characteristics, but approximately the same order. The codes of Galois fields elements are represented in a polynomial basis.

2 Hardware Implementation of Cryptographic Protection Units in FPGA

To estimate the hardware costs of multipliers for various Galois fields, an automated system (core generator) for configuration of data protection unit models was developed.

The purpose of the system is to form a VHDL-description of a necessary unit with necessary characteristics. To accomplish this task, the generator uses the base configurable unit model in the form of its VHDL-description (template) [4].

The generator determines which parts of the base model will be needed for a selected unit and performs adjustment of corresponding parameters of each selected element of the base model. A word processor was developed for VHDL-code generation from the base model of the configured unit.

Configuration options include the type of unit, field characteristic p, and field order m.

The list of cells that can be generated includes modified Guild cells for the field with characteristic p, carry generation cells F, as well as the actual multiplier.

The word processor generates VHDL-description in accordance with the template, taking into account the parameters of the unit. The VHDL-generator of the Galois field multipliers implements 2 variants of the multipliers: in the first variant the MGC is treated as an integer component (black box, read only memory, ROM) and in the second variant the MGC consists of a multiplier and an adder.

The generator of VHDL-descriptions of multipliers consists of the following parts (Fig. 1):

- module for inverter F creation;
- module for multiplier MUL and adder SUM creation;
- module for element MGC creation;
- module for creating and filling the matrix with F elements and the links between them;
- module for creating and filling the matrix with MGC elements and the links between them.

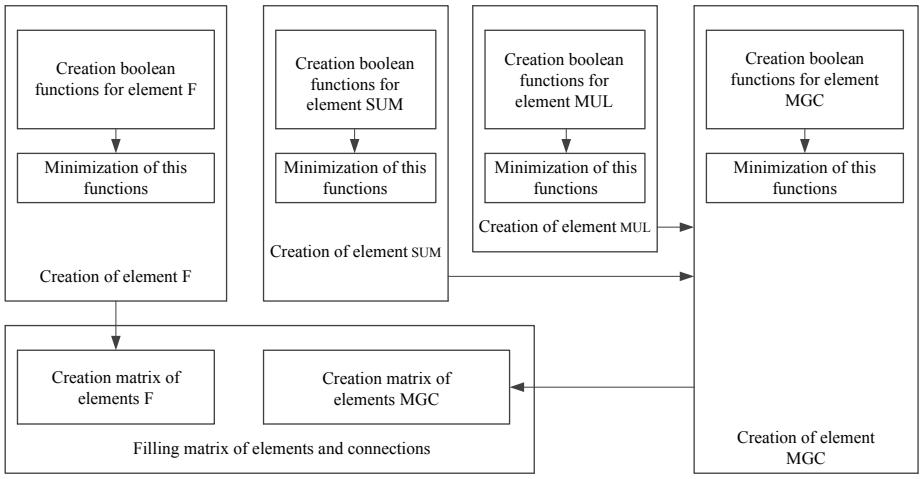

Fig. 1. Structural diagram of VHDL-descriptions generator of Galois field GF(p^m) elements multipliers

The program modules which are responsible for creating the F and MGC elements for both cases use the common part of the program to minimize logical functions.

Intermediate results of the VHDL-descriptions generator are stored in files. This allows the user to see what changes occur with the function at each stage.

In Fig. 2 the internal structure of the MGC is presented in case of its implementation a) as a "black box" and b) as modular multiplier and adder, on the example of the multiplier for Galois Field GF(3^4).

In Fig. 3 the internal structure of generated multiplier for Galois Field GF(3^4) is presented. The inverter F performs operation B=(-A)modp.

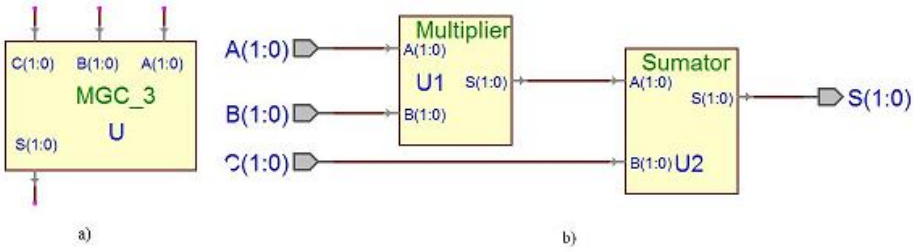

Fig. 2. Implementation of MGC for Galois fields GF(3^m): a) as "black box" (BB); b) as “Multiplier plus Adder” (MA).

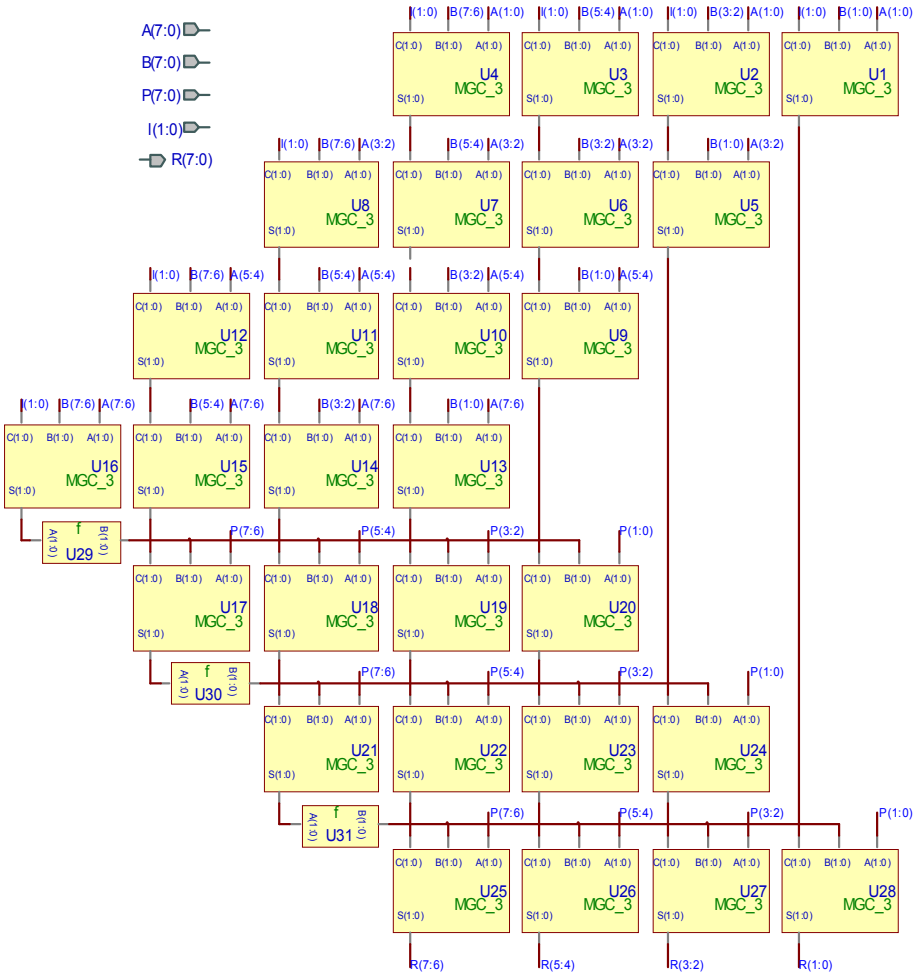

Fig. 3. Diagram of Galois Field GF(3^4) multiplier MUL (U1 – U16 – multiplier, U17 – U31 - modulo convolution unit, divider)

In the first case, two logical functions S(1) and S(0) which depend on 6 variables (C(1:0), B(1:0), A(1:0)) are formed. The function S(1:0) forms the result of S = (A * B + C)mod3. In the second variant there are 4 logical functions, each of which depends on 4 variables, two of which form the result of multiplication F = A * Bmod3, and two others - result of adding S = (F + C) mod3.

The projects in this work were created and simulated in the Active-HDL 9.1 environment. The implementation was performed in the Xilinx ISE environment for Spartan 6 FPGA.

The value of hardware costs and time delays for the implementation of the multipliers for GF(2^{15}), GF(3^{9}), GF(5^{6}), GF(7^{5}), GF(13^{4}) fields, which all have schemes similar to Fig. 3, are shown in Fig. 4 and Table 1.

With the graphics of Fig. 4, it can be seen that multiplier for GF(2^{15}) has the smallest hardware cost. Multipliers for field GF(2^{15}) has also the smallest time delays. It should also be noted that the multipliers for GF(3^{9}), GF(7^{5}) have hardware and performance rates, which are somewhat higher than logical fields multipliers.

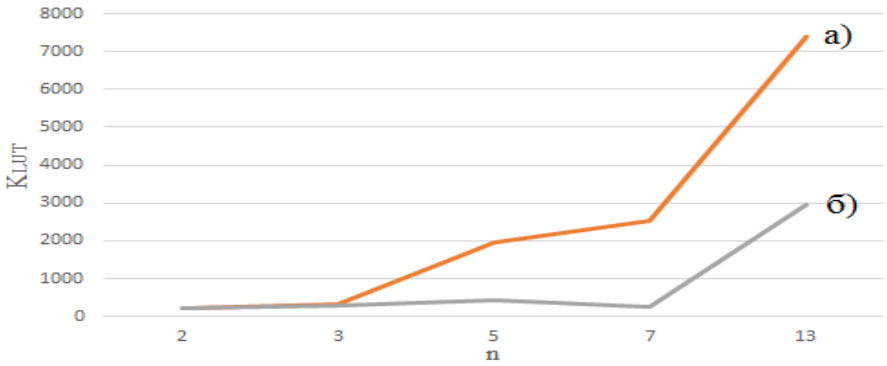

Fig. 4. Hardware costs of the Galois Field GF(2^{15}), GF(3^{9}), GF(5^{6}), GF(7^{5}), GF (13^{4}) multipliers for Spartan 6 FPGA: a) Multiplier plus Adder; b) "black box".

3 Software Implementation of Cryptographic Data Protection Units

3.1 Software Complexity.

The hacking of cryptographic cipher on the basis of elliptic curves is mostly realized by the method of "brute force". To assess the stability of the cipher to breakdown, it is necessary to analyze the software complexity of arithmetic operations in the Galois fields. The most complicated arithmetic operation in the Galois fields GF(pm) is multiplication. For analysis, it was assumed that the multiplication operation is performed on a matrix multiplier. Matrix multiplier consists of modified Guild cells. The modified Guild cell can be considered either as a holistic element or as a set of multipliers and adder both by the modulus of the characteristic p of the field. The

software complexity of multipliers in different Galois fields will be determined by the number of logical operations that must be performed to calculate 1 bit of the result.

Table 1. Hardware costs (LUTs and slices) of Galois Field multipliers for Spartan 6 FPGA.

Field	MGS as	MGC number	Number of GF elements, %	Number of LUTs in multipliers	Number of slices in multipliers	Number of inputs and outputs	max delay, ns
$GF(2^{15})$	BB	435	101,3%	218	82	61	22
$GF(2^{15})$	MA	435	101,3%	205	86	61	26
$GF(3^{9})$	BB	153	96,5%	312	138	74	31
$GF(3^{9})$	MA	153	96,5%	298	106	74	47
$GF(5^{6})$	BB	66	89,6%	1946	600	75	49
$GF(5^{6})$	MA	66	89,6%	439	171	75	42
$GF(7^{5})$	BB	45	95,4%	2534	963	63	37
$GF(7^{5})$	MA	45	95,4%	258	103	63	31
$GF(13^{4})$	BB	28	100%	7395	3031	68	42
$GF(13^{4})$	MA	28	100%	2949	1018	68	86

The Table 1 shows that the smallest hardware costs of the multiplier will be in the $GF(2^{15})$ field. Fields with characteristic 3 and 7 have higher hardware costs, respectively, of 45.36 and 25.85% for MGC as a multiplier and adder

Table 2 shows the number of logical operations that need to be performed to simulate one modified Guild cell for fields with different characteristics. For analysis, fields with approximately the same number of elements are taken. The values given in Table 2 are based on the results of the synthesis of the corresponding MGS created by the core generator of cryptographic data protection units.

Table 2. The number of logical elements for creating MGC.

Galois field $GF(p^m)$	MGC is holistic element	MGC consists from multiplier and adder	The largest length of the MGC column (Fig. 3)
$GF(2^{100})$	2	2	199
$GF(3^{63})$	28	30	125
$GF(5^{43})$	380	137	85
$GF(7^{35})$	440	129	69
$GF(13^{27})$	3124	870	53

3.2 The First Method of Software Complexity Estimation.

To estimate the number of operations that need to be performed when modeling a MGC, its model is used in the form of a multiplier plus adder (consists of two ROMs, Fig. 2, b – the first mode of estimation). An option when the MGC is holistic element is not considered, since its software implementation needs in general more operations, as can be seen from Table 2.

Each of MGC's two elements can be imagined as a ROM, so the hardware complexity $O_{MCG}(p)$ of the MGC is the volume of this two ROMs: $O_{MCG}(p) = 2V(p)=2*2^{ni}*no,$ where $ni = 2\lceil log_2 p \rceil$ is ROM input bits number and $no = \lceil log_2 p \rceil$ is ROM output bits number.

Then $O_{MCG}(p) = 2 * 2^{2\lceil log_2 p \rceil} \lceil log_2 p \rceil = 2^{2\lceil log_2 p \rceil + 1} \lceil log_2 p \rceil.$

The number N_{MGC} of MGC in the Galois fields $GF(p^m)$ multiplier is $N_{MGC}(m) = m(2m-1)$. The full hardware complexity of the multiplier of the Galois field $GF(p^m)$ elements can be calculated as $O_{MUL}(m,p) = O_{MCG}(p) N_{MGC}(m) = 2^{2\lceil log_2 p \rceil + 1} \lceil log_2 p \rceil m(2m-1)$. This value shows total number of bits which have to be calculated during multiplication.

Table 3 shows the hardware complexity of $GF(p^m)$ multipliers.

Table 3. Hardware complexity of $GF(p^m)$ multipliers.

Galois field $GF(p^m)$	Hardware complexity	Relative hardware complexity	NC	NV	Software complexity	Relative software complexity
$GF(2^{100})$	159200	1,00	8	16	1244	1,00
$GF(3^{63})$	20800	0,13	4	16	325	0,26
$GF(5^{43})$	1403520	8,82	3	16	29240	23,51
$GF(7^{35})$	927360	5,83	3	16	19320	15,53
$GF(13^{27})$	2930688	18,41	2	16	91584	73,64

The software complexity of $GF(p^m)$ multipliers can be estimated as $O_{SW}(m,p) = O_{MUL}(m,p) / NC \cdot NV$, where NC is number of processor cores; NV is the number of vectors that the processor core can handle [7]. This value shows how many Galois Field elements each core can handle at a time. The element code length must be equal or less then vector length $VL \geq \lceil log_2 p \rceil$ and $NC \cdot NV \geq m$.

The number of vectors NV and their length VL in modern processors are given in Table 4 [8]. Results of software complexity estimation are shown in Table 3. This mode of software complexity estimation gives incorrect results especially in case of Galois fields with big characteristics (Fig. 5).

Table 4. The number of vectors and their length.

Vector length, bit (VL)	The number of vectors (NV)
8	8, 16
16	4, 8
32	2, 4, 16
64	2, 8

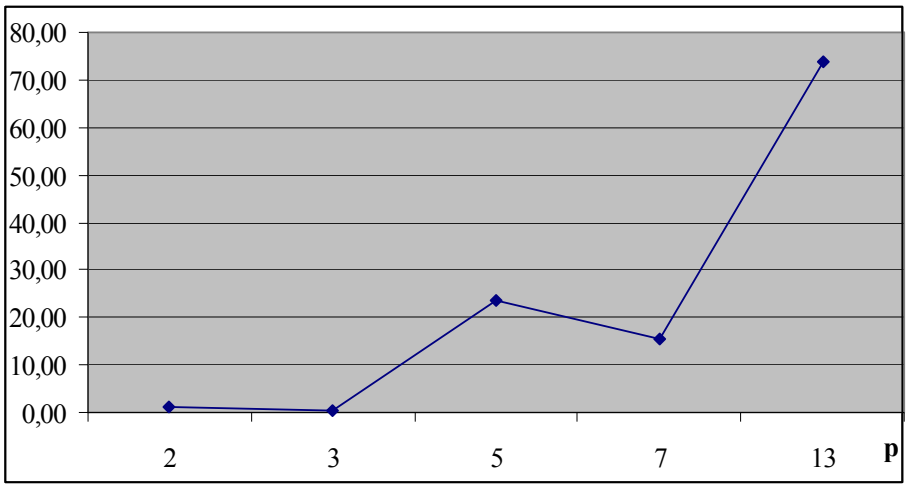

Fig. 5. The relative software complexity of multiplication (the first method).

3.3 The Second Method of Software Complexity Estimation.

The software complexity of multiplication can also be estimated on the basis of arithmetic operations that can be performed in each MGC (the second mode of estimation). In one MGC, you must perform a multiplication, division, addition and re-division operation. This method can be implemented by the MGC for the Galois field. Each MGC will perform 4 arithmetic operations, which have their weight - the ratio of its execution time to execution time of logical operation. Take the weight of multiplication and division operation as 8, the weight of addition as 4. Thus, for the simulation of the work of the MGC, it is necessary to perform 4 arithmetic operations, which are equivalent to 28 logical operations. Estimation results are in Table 5 and in Fig. 6.

3.4 The Third Method of Software Complexity Estimation.

An estimation of software complexity of the multiplier on the basis of logical operations used for multiplication, division, addition is described below (the third estimation method). To estimate the complexity of the multiplier it is necessary to

calculate the complexity of the MGC. It performs Nbpm=$\lceil log_2 p \rceil$ bit-parallel multiplication operations, Nbprm=$\lceil log_2 p \rceil$ bit-parallel operations of reduction in modulus after multiplication (long division), Na=1 serial addition operation and Nbpra=1 bit-parallel operations of reduction in modulus after addition (short division). If we take that bit-parallel and serial operation have the same execution time then the complexity of the MGC can be estimated as total quantity Nbps of such operations Nbps = Nbpm + Nbprm + Na + Nbpra = $= 2\lceil log_2 p \rceil + 2 = 2(\lceil log_2 p \rceil + 1)$. The scheme of the multiplier and divider is shown in Fig. 3. In general, for the implementation of the MGC, it is necessary to perform Nlo = $4 \cdot 2(\lceil log_2 p \rceil + 1) = 8(\lceil log_2 p \rceil + 1)$ logical operations, since we take that each multiplier and divider performs approximately 4 logical operations, for the implementation of a field with a characteristic of no more than W (where W is width of processor data bus), and $NLO = 8(\lceil log_2 p \rceil + 1)\lceil \lceil log_2 p \rceil / W \rceil$ for a field with any other characteristic. MGC number is NMGC=m(2m-1). The formula for estimating the number of logical operations that is used to multiply 2 elements of a field: $NLOM = 8(\lceil log_2 p \rceil + 1)\lceil \lceil log_2 p \rceil m(2m-1)/W \rceil$. Estimation results are in Table 6 and in Fig. 6.

Table 5. The number of logical operations for the 2nd estimation method.

Galois fields $GF(p^m)$	The total number of logical operations that must be performed for MGC	The number of MGC in the multiplier	The total number of logical operations that must be performed to multiply the elements of the field	Relative software complexity
$GF(2^{100})$	28	19900	557200	1
$GF(3^{63})$	28	7875	220500	0,39
$GF(5^{43})$	28	3655	102340	0,18
$GF(7^{35})$	28	2415	67620	0,12
$GF(13^{27})$	28	1431	40068	0,07

The results of the comparison of software complexity are shown in Fig. 6.

Table 6. The number of logic operations for the third method.

Galois field $GF(p^m)$	The total number of logical operations that must be performed for MGC	The number of MGC in the multiplier	The total number of logical operations that must be performed to multiply the elements of the field	Relative software complexity
$GF(2^{100})$	16	19900	318400	1
$GF(3^{63})$	28	7875	220500	0,69
$GF(5^{43})$	40	3655	146200	0,45
$GF(7^{35})$	40	2415	96600	0,3
$GF(13^{27})$	52	1431	74412	0,23

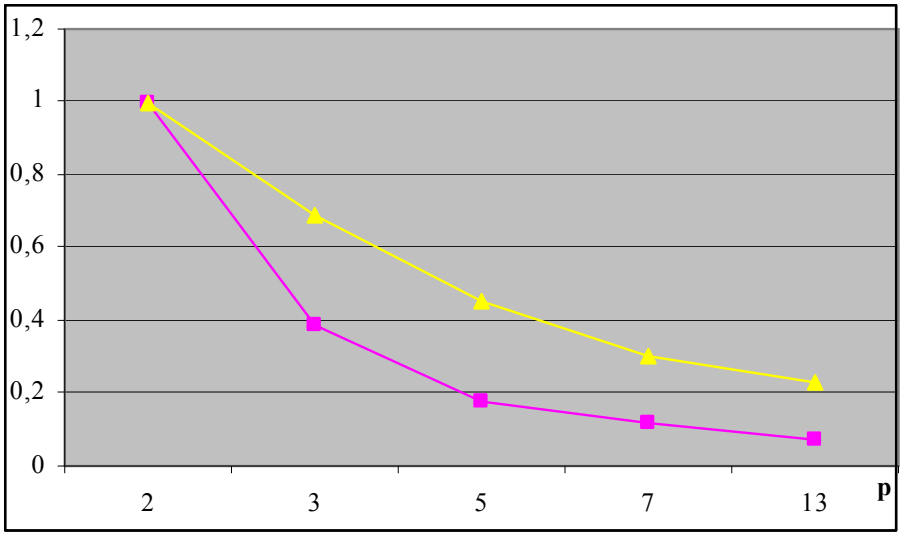

Fig. 6. Comparison of software complexity (squares, triangles – 2nd and 3rd estimation methods).

4 Results

The obtained results show that the modeling of the multiplier of the Galois field elements is better by the method of logical operations.

The software complexity of multiplier for a field with a small characteristics is the largest. That is, it will be harder to crack them.

At the same time, the analysis of hardware implementation of multipliers shows that the hardware complexity of multipliers for fields with characteristics 2, 3 and 7 is the smallest.

Thus, the use of hardware multipliers in Galois fields with characteristics 2, 3, and 7 provides the best hardware parameters and complicates the task of modeling their work by hackers.

5 Conclusion

An automated system for configuring VHDL-descriptions of cryptographic data protection units has been developed. With its help, a family of multipliers of the Galois field elements was generated for the Galois fields with field characteristics 2, 3, 5 7, 13 and a hardware cost analysis for their implementation on the FPGA of was performed. An analysis of the results of multiplier implementation has shown that the least hardware costs will be in multipliers of the Galois fields with the field characteristic 2. It is also shown that extended fields with approximately the same number of elements and small characteristics have low hardware and high software complexity.

6 References

1. I. Zholubak, V. Hlukhov. Determination of the extended Galois field GF(d^m) with the smallest hardware complexity of the multiplier // Bulletin of the National Polytechnic University “Information systems and network” ”. – Lviv: - 2016. - № 835. - P. 50 - 58.
2. I. Zholubak, V. Hlukhov. Hardware costs of Galois multipliers GF(d^m) with a large characteristic // Bulletin of the National Polytechnic University “Computer sciences and information technology” ”. – Lviv: - 2017.
3. Cherkasky M., Tkachuk T. I. Characteristics of the complexity of multiplication devices // Radioelectronic and computer systems. - 2012. - № 5. P. 142 – 147 .
4. Melnik A., Melnik V. Personal supercomputers. Monograph. Lviv: Lviv Polytechnic Publishing House, 2013. 516 p.
5. I. Zholubak, V. Hlukhov. Implementation of high-order Galois multipliers in FPGA // Bulletin of the National Polytechnic University “Computer systems and network”. – Lviv: - 2017. - № 881. - 3. 41 - 47.
6. Palagin A. The structure of FPGA-based cyclic-code converters / Alexander Palagin, Vladimir Opanasenko, Sergey Krivoi // Optical Memory & Neural Networks (Information Optics). Springer. – 2013, Vol. 22, N.4. – PP. 207–216.
7. Pawliczek P., Dzwinel W., Yuen D.A. (2015) Visual Exploration of Data with Multithread MIC Computer Architectures. In: Rutkowski L., Korytkowski M., Scherer R., Tadeusiewicz R., Zadeh L., Zurada J. (eds) Artificial Intelligence and Soft Computing. ICAISC 2015. Lecture Notes in Computer Science, vol 9120. Springer, Cham
8. Stephen Blair-Chappell. The significance of SIMD, SSE and AVX. Software and Services Group. Copyright © 2010. Intel Corporation.
9. Vassyl Dimitrov, Kimmo Järvinen, "Another Look at Inversions over Binary Fields" in Computer Arithmetic (ARITH), 2013 21st IEEE Symposium, Austin, TX, USA, 2013.