*Communications in Applied Analysis*, **25**, No. 1 (2021), 23-37 ISSN: 1083-2564

# A REFINEMENT OF THE KNUTH'S EXTENDED EUCLIDEAN ALGORITHM FOR COMPUTING MODULAR MULTIPLICATIVE INVERSE

Anton Iliev[1,2], Nikolay Kyurkchiev[1,2], and Asen Rahnev[1]

[1]Faculty of Mathematics and Informatics
University of Plovdiv Paisii Hilendarski
24, Tzar Asen Str., 4000 Plovdiv, BULGARIA

[2]Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8, 1113 Sofia, BULGARIA

**ABSTRACT:** In many papers [10]–[44] we present effective iterative and recursive schemes for the all classes of algorithms that concern the classical task of finding greatest common divisor. Here we present a new algorithm for finding modular multiplicative inverse, which is based on combination of "remainder" and "difference" operations. The approach presented here is based on adaptation of the Knuth's extended Euclidean algorithm. The main reason of this research interest is because when we are dealing with long numbers the CPU usage for "remainder" operation leads to slower calculation process.

**AMS Subject Classification:** 11A05, 68W01
**Key Words:** modular multiplicative inverse, reduced number of operations

**Received:** March 22, 2021 ; **Accepted:** May 15, 2021 ;
**Published:** May 18, 2021. **doi:** 10.12732/caa.v25i1.3
Dynamic Publishers, Inc., Acad. Publishers, Ltd. http://www.acadsol.eu/caa

## 1. INTRODUCTION

Let $a$ and $b$ be a natural numbers. We set our task to make iteration process, which compute recursively and iteratively the modular multiplicative inverse of $a$ modulo $b, a^{-1}$ ( mod $b$). The computational process will return as result the inverse as natural number less than $b$ or 0 if no inverse exists. The multiplicative inverse of $a$ modulo $b$ exists if and only if their greatest common divisor ($gcd$) = 1. We use also optimization which is that when $a$ and $b$ are even numbers we can skip computational process because $gcd \geq 2$. For the theoretical importance of the task, see [1]–[6] and [45]–[58]. Some recent applications of this class of algorithms are presented in [53]–[57].

For testing purposes we will use the following computer: processor – Intel(R) Core(TM) i7-6700HQ CPU 2.60GHz, 2592 Mhz, 4 Core(s), 8 Logical Processor(s), RAM 16 GB, Microsoft Windows 10 Enterprise x64, Microsoft Visual C# 2017 x64.

We will note that in our previous studies we received highly optimized and symmetrized algorithms [12] using mainly "remainder" operation, which is extremely fast for regular integers. Namely, iterative algorithm is –

**Algorithm 1.**

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi = 0;
else {
b0 = b; iter = 1; x1 = 1; x2 = 0;
if (a > b)
do
{
q = a / b; a %= b; t = x1 + q * x2; x1 = x2; x2 = t; iter = -iter;
if (a < 1)
{
if (b > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else if (iter < 0) eeacmi = b0 - x1; else eeacmi = x1; break;
}
q = b / a; b %= a; t = x1 + q * x2; x1 = x2; x2 = t; iter = -iter;
if (b < 1)
```

```
{
if (a > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else if (iter < 0) eeacmi = b0 - x1; else eeacmi = x1; break;
}
} while (true);
else
do
{
q = b / a; b %= a; t = x2 + q * x1; x2 = x1; x1 = t; iter = -iter;
if (b < 1)
{
if (a > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = x2; else eeacmi = b0 - x2; break;
}
q = a / b; a %= b; t = x2 + q * x1; x2 = x1; x1 = t; iter = -iter;
if (a < 1)
{
if (b > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = x2; else eeacmi = b0 - x2; break;
}
} while (true);
}
```

and its recursive presentation is –

**Algorithm 2.**

```
static long Euclid(long a, long b, ref long x, ref long y, ref long iter)
{
long r = a % b; long q1 = a / b; iter = -iter;
if (r < 1) { x = 1; y = 0; return b; }
long u = b % r; long q2 = b / r; iter = -iter;
if (u < 1) { x = q1; y = 1; return r; }
```

```
long d = Euclid(r, u, ref x, ref y, ref iter);
y += q2 * x; x += q1 * y;
return d;
}
```

Using programming statements in "Numerical Example" of this paper the recursive algorithm 2 above can be called by:

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi = 0;
else {
iter = 1;
gcd = Euclid(a, b, ref y, ref x, ref iter);
if (gcd > 1 ||(x == 0 && y == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = b - x; else eeacmi = x;
}
```

In our recent paper [35], we optimize the binary algorithm for computing modular multiplicative inverse from [52] using another boundary condition in main loop (in our algorithm it is u!=v instead of u!=0 in original algorithm) as well as we reorganize the other parts of original algorithm with leading purpose to reduce some unnecessary operations for the faster final result obtaining. Both algorithms given in [35] and [52] can be used for simultaneously finding of $eeacmi1 := a^{-1} \pmod{b}$ as well as $eeacmi2 := b^{-1} \pmod{a}$. As an example here we give iterative version of optimized algorithm from [35]:

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi1 = eeacmi2 = 0;
else
{
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
u = a; v = b;
while ((u & 1) == 0)
{
u >>= 1;
if ((x1 & 1) == 0 && (x2 & 1) == 0) { x1 >>= 1; x2 >>= 1; }
else { x1 = (x1 + b) >> 1; x2 = (x2 - a) >> 1; }
```

```
}
while ((v & 1) == 0)
{
v >>= 1;
if ((y1 & 1) == 0 && (y2 & 1) == 0) { y1 >>= 1; y2 >>= 1; }
else { y1 = (y1 + b) >> 1; y2 = (y2 - a) >> 1; }
}
while (u != v)
if (u > v)
{
u -= v; x1 -= y1; x2 -= y2;
do
{
u >>= 1;
if ((x1 & 1) == 0 && (x2 & 1) == 0) { x1 >>= 1; x2 >>= 1; }
else { x1 = (x1 + b) >> 1; x2 = (x2 - a) >> 1; }
} while ((u & 1) == 0);
}
else
{
v -= u; y1 -= x1; y2 -= x2;
do
{
v >>= 1;
if ((y1 & 1) == 0 && (y2 & 1) == 0) { y1 >>= 1; y2 >>= 1; }
else { y1 = (y1 + b) >> 1; y2 = (y2 - a) >> 1; }
} while ((v & 1) == 0);
}
if (v > 1) eeacmi1 = eeacmi2 = 0;
else
{
while (y1 < 0) y1 += b; while (y1 >= b) y1 -= b; eeacmi1 = y1;
while (y2 < 0) y2 += a; while (y2 >= a) y2 -= a; eeacmi2 = y2;
}
}
```

as well as the algorithm from [35] which mainly uses in its processing part "difference" operation

```
b0 = b; a0 = a;
if ((a & 1) == 0 && (b & 1) == 0) eeacmi1 = eeacmi2 = 0;
else
{
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
while (a != b)
if (a > b) { a -= b; x1 -= y1; x2 -= y2; }
else { b -= a; y1 -= x1; y2 -= x2; }
if (a > 1)
eeacmi1 = eeacmi2 = 0;
else
{
while (y1 < 0) y1 += b0; while (y1 >= b0) y1 -= b0; eeacmi1 = y1;
while (y2 < 0) y2 += a0; while (y2 >= a0) y2 -= a0; eeacmi2 = y2;
}
}
```

**2. Main Results.**

We propose the following:

**Algorithm 3.**

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi = 0;
else {
b0 = b; iter = 1; x1 = 1; x2 = 0;
do
if (a > b)
{ q = a / b; a %= b; t = x1 + q * x2; x1 = x2; x2 = t; iter = -iter;
if (a < 1)
{ if (b > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = b0 - x1; else eeacmi = x1; break; }
```

```
b -= a; t = x1 + x2; x1 = x2; x2 = t; iter = -iter;
if (a == b)
{ if (b > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = b0 - x1; else eeacmi = x1; break; } }
else
{ q = b / a; b %= a; t = x2 + q * x1; x2 = x1; x1 = t; iter = -iter;
if (b < 1)
{ if (a > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = x2; else eeacmi = b0 - x2; break; }
a -= b; t = x2 + x1; x2 = x1; x1 = t; iter = -iter;
if (a == b)
{ if (a > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = x2; else eeacmi = b0 - x2; break; } }
while (true);
}
```

and its recursive implementation as

**Algorithm 4.**

```
static long Euclid(long a, long b, ref long x,
ref long y, ref long iter)
{
long r = a % b; long q1 = a / b; iter = -iter;
if (r < 1) { x = 1; y = 0; return b; }
long u = b - r; iter = -iter;
if (u == r) { x = q1; y = 1; return r; }
long d;
if (u > r) { iter = -iter; d = Euclid(u, r, ref y, ref x, ref iter); }
else d = Euclid(r, u, ref x, ref y, ref iter);
y += x; x += q1 * y;
return d;
```

```
}
```

**Numerical Example.**

We will compare the new algorithms 3 and 4 with classical adaptations of the Knuth's extended Euclidean iterative and recursive algorithms [45], and the new originally introduced by us algorithms 1 and 2 [12].

Classical adaptation of the Knuth's extended Euclidean iterative implementation [45], [12] is

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi = 0;
else {
b0 = b; x1 = 1; x2 = 0; iter = 1;
while (b > 0)
{
q = a / b; t1 = a % b; t = x1 + q * x2;
x1 = x2; x2 = t; a = b; b = t1; iter = -iter;
}
if (a > 1 ||(x1 == 0 && x2 == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = b0 - x1; else eeacmi = x1;
}
```

and its recursive variant

```
static long Euclid(long a, long b, ref long x,
ref long y, ref long iter)
{
if (b < 1) { x = 1; y = 0; return a; }
long q = a / b; long r = a % b; iter = -iter;
long d = Euclid(b, r, ref y, ref x, ref iter);
y += q * x;
return d;
}
```

can be called by:

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi = 0;
else {
iter = 1;
gcd = Euclid(a, b, ref x, ref y, ref iter);
if (gcd > 1 ||(x == 0 && y == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = b - x; else eeacmi = x;
}
```

Algorithm 4. from present paper can be called by:

```
if ((a & 1) == 0 && (b & 1) == 0) eeacmi = 0;
else {
iter = 1;
gcd = Euclid(a, b, ref y, ref x, ref iter);
if (gcd > 1 ||(x == 0 && y == 1)) eeacmi = 0;
else
if (iter < 0) eeacmi = b - x; else eeacmi = x;
}
```

Below is the source code of the main function, which we have used for our testing purposes:

```
long a, b, t, t1, q, eeacmi, iter, d = 0, eeacmi1, eeacmi2, d1 = 0, d2 = 0;
long gcd, b0, a0, x1, x2, x = 0, y = 0, y1, y2, u, v;
for (int i = 1; i < 100000001; i++) { a = i; b = 200000002 - i;
//here is the source code of every one of algorithms 1, 3
//classical adaptation of the Knuth's extended Euclidean iterative
// and both binary algorithms for computing modular
// multiplicative inverse (from Introduction of this paper),
// and calling of algorithms 2, 4 and
//classical adaptation of the Knuth's extended Euclidean recursive
d += eeacmi; d1 += eeacmi1; d2 += eeacmi2;
}
```

```
Console.WriteLine(d); Console.WriteLine(d1); Console.WriteLine(d2);
```

CPU time results are:

CPU time of Algorithm 1 is: **15.713 seconds.**
CPU time of Algorithm 3 is: **16.947 seconds.**
CPU time of classical adaptation of the Knuth's extended Euclidean iterative is: **18.148 seconds.**

CPU time of Algorithm 2 is: **24.509 seconds.**
CPU time of Algorithm 4 is: **28.857 seconds.**
CPU time of classical adaptation of the Knuth's extended Euclidean recursive is: **32.810 seconds.**

## 2. CONCLUSION

Again we confirm that presented algorithms 3 and 4 are in Strassen [58] style by decreasing number of "remainder" operations and increasing quantity of "difference" operations.

## ACKNOWLEDGEMENTS

This work has been accomplished with the financial support by the Grant No BG05M2OP001-1.001-0003, financed by the Science and Education for Smart Growth Operational Program (2014-2020) and co-financed by the European Union through the European structural and Investment funds.

## REFERENCES

[1] A. Akritas, A new method for computing polynomial greatest common divisors and polynomial remainder sequences, *Numerische Mathematik*, **52** (1988), 119–127.

[2] S. Enkov, *Programming in Arduino Environment*, University Press "Paisii Hilendarski", Plovdiv (2017). (in Bulgarian)

[3] F. Chang, Factoring a Polynomial with Multiple-Roots, *World Academy of Science, Engineering and Technology*, **47** (2008), 492–495.

[4] Th. Cormen, Ch. Leiserson, R. Rivest, Cl. Stein, *Introduction to Algorithms*, 3rd ed., The MIT Press, Cambridge (2009).

[5] K. Garov, A. Rahnev, *Textbook-notes on programming in BASIC for facultative training in mathematics for 9.–10. Grade of ESPU*, Sofia (1986). (in Bulgarian)

[6] A. Golev, *Textbook on algorithms and programs in C#*, University Press "Paisii Hilendarski", Plovdiv (2012). (in Bulgarian)

[7] T. Terzieva, *Introduction to web programming*, University Press "Paisii Hilendarski", Plovdiv (2021), ISBN 978-619-202-623-3. (in Bulgarian)

[8] T. Terzieva, *Development of algorithmic thinking in the Informatics Education*, University Press "Paisii Hilendarski", Plovdiv (2021), ISBN 978-619-202-622-6. (in Bulgarian)

[9] T. Terzieva, *Educational tools for teaching in digital environment*, University Press "Paisii Hilendarski", Plovdiv (2021). (in Bulgarian)

[10] A. Iliev, N. Kyurkchiev, A Note on Knuth's Implementation of Euclid's Greatest Common Divisor Algorithm, *International Journal of Pure and Applied Mathematics*, **117** (2017), 603–608.

[11] A. Iliev, N. Kyurkchiev, A. Golev, A Note on Knuth's Implementation of Extended Euclidean Greatest Common Divisor Algorithm, *International Journal of Pure and Applied Mathematics*, **118** (2018), 31–37.

[12] A. Iliev, N. Kyurkchiev, A. Rahnev, A Note on Adaptation of the Knuth's Extended Euclidean Algorithm for Computing Multiplicative Inverse, *International Journal of Pure and Applied Mathematics*, **118** (2018), 281–290.

[13] A. Iliev, N. Kyurkchiev, A Note on Euclidean and Extended Euclidean Algorithms for Greatest Common Divisor for Polynomials, *International Journal of Pure and Applied Mathematics*, **118** (2018), 713–721.

[14] A. Iliev, N. Kyurkchiev, A Note on Least Absolute Remainder Euclidean Algorithm for Greatest Common Divisor, *International Journal of Scientific Engineering and Applied Science*, **4** No. 3 (2018), 31–34.

[15] A. Iliev, N. Kyurkchiev, A Note on Knuth's Algorithm for Computing Extended Greatest Common Divisor using SGN Function, *International Journal of Scientific Engineering and Applied Science*, **4** No. 3 (2018), 26–29.

[16] A. Iliev, N. Kyurkchiev, *New Trends in Practical Algorithms: Some Computational and Approximation Aspects*, LAP LAMBERT Academic Publishing, Beau Bassin (2018).

[17] A. Iliev, N. Kyurkchiev, 80th Anniversary of the birth of Prof. Donald Knuth, *Biomath Communications*, **5** (2018), 7 pp.

[18] A. Iliev, N. Kyurkchiev, New Realization of the Euclidean Algorithm, *Collection of scientific works of Eleventh National Conference with International Participation Education and Research in the Information Society*, Plovdiv, ADIS, June 1–2, (2018), 180–185. (in Bulgarian)

[19] A. Iliev, N. Kyurkchiev, New Organizing of the Euclid's Algorithm and one of its Applications to the Continued Fractions, *Collection of scientific works from conference "Mathematics. Informatics. Information Technologies. Application in Education"*, Pamporovo, Bulgaria, 10–12 October 2018, (2019), 199–207.

[20] A. Iliev, N. Kyurkchiev, The faster Euclidean algorithm, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 15–20.

[21] A. Iliev, N. Kyurkchiev, The faster extended Euclidean algorithm, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 21–26.

[22] P. Kyurkchiev, V. Matanski, The faster Euclidean algorithm for computing polynomial multiplicative inverse, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 43–48.

[23] V. Matanski, P. Kyurkchiev, The faster Lehmer's greatest common divisor algorithm, *Collection of scientific works from conference*, Pamporovo, Bulgaria, 28–30 November 2018, (2019), 37–42.

[24] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement Euclidean Algorithm for Greatest Common Divisor. I, *Neural, Parallel, and Scientific Computations*, **26** No. 3 (2018), 355–362.

[25] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Harris–Stein Modification of Euclidean Algorithm for Greatest Common Divisor. II, *International Journal of Pure and Applied Mathematics*, **120** No. 3 (2018), 379–388.

[26] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Least Absolute Remainder Algorithm for Greatest Common Divisor. III, *Neural, Parallel, and Scientific Computations*, **27** No. 1 (2019), 1–9.

[27] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Tembhurne–Sathe Modification of Euclidean Algorithm for Greatest Common Divisor. IV, *Dynamic Systems and Applications*, **28** No. 1 (2019), 143–152.

[28] A. Iliev, N. Kyurkchiev, A. Rahnev, *Nontrivial Practical Algorithms: Part 2*, LAP LAMBERT Academic Publishing, Beau Bassin (2019).

[29] A. Iliev, N. Valchanov, T. Terzieva, Generalization and Optimization of Some Algorithms, *Collection of scientific works of National Conference "Education in Information Society"*, Plovdiv, ADIS, 12–13 May 2009, (2009), 52–58. (in Bulgarian)

[30] H. Gyulyustan, A Note on Euclidean Sequencing Algorithm, *Proceedings of the Scientific Conference "Innovative ICT for Digital Research Area in Mathematics, Informatics and Pedagogy of Education"*, Pamporovo, 7–8 November 2019, Plovdiv University Press, (2020), 57–64.

[31] A. Iliev, N. Kyurkchiev, A. Rahnev, New Algorithm for Finding Greatest Common Divisor, *Neural, Parallel, and Scientific Computations*, 28 No. 1 (2020), 69–74.

[32] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Stein's Binary Algorithm for Finding Greatest Common Divisor, *Neural, Parallel, and Scientific Computations*, 28 No. 1 (2020), 75–80.

[33] A. Iliev, N. Kyurkchiev, A. Rahnev, New Algorithms for Finding Modular Multiplicative Inverse, *Neural, Parallel, and Scientific Computations*, 28 No. 1 (2020), 81–88.

[34] A. Iliev, N. Kyurkchiev, A. Rahnev, New Extended Algorithm for Finding Greatest Common Divisor, *Neural, Parallel, and Scientific Computations*, 28 No. 1 (2020), 89–95.

[35] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Modular Multiplicative Inverse Binary Algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 14 No. 1 (2020), 37–44.

[36] A. Iliev, N. Kyurkchiev, A. Rahnev, Recursive Extended Stein's Binary Algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 14 No. 1 (2020), 31–36.

[37] A. Iliev, N. Kyurkchiev, A. Rahnev, A new improvement of Jacobi symbol algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 15 No. 1 (2020), 13–22.

[38] A. Iliev, N. Kyurkchiev, A. Rahnev, A new improvement of Jacobi symbol binary algorithm, *International Electronic Journal of Pure and Applied Mathematics*, 15 No. 1 (2020), 1–11.

[39] A. Iliev, N. Kyurkchiev, A. Rahnev, Efficient Binary Algorithm for Kronecker Symbol, *Communications in Applied Analysis*, 25 No. 1 (2021), 11–21.

[40] A. Iliev, N. Kyurkchiev, A. Rahnev, Efficient Algorithm for Kronecker Symbol, *International Electronic Journal of Pure and Applied Mathematics*, 15 No. 1 (2020), 23–30.

[41] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, A Refinement of the Extended Euclidean Algorithm using SGN Function, (2021), preprint.

[42] A. Iliev, N. Kyurkchiev, A. Rahnev, A Refinement of the Extended Euclidean Algorithm, (2021), preprint.

[43] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, A Refinement of the Böhs Algorithm for Computing Modular Multiplicative Inverse, (2021), preprint.

[44] A. Iliev, N. Kyurkchiev, A. Rahnev, A New Improvement of Extended Stein's Binary Algorithm, *Proceedings of the Anniversary International Scientific Conference Synergetics and Reflection in Mathematics Education*, Pamporovo, 16–18 October 2020, Plovdiv University Press, (2020), 259–264.

[45] D. Knuth, *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Boston (1998).

[46] Hr. Krushkov, A. Iliev, *Practical programming guide in Pascal, Parts I and II*, Koala press, Plovdiv (2002). (in Bulgarian)

[47] P. Nakov, P. Dobrikov, *Programming=++Algorithms*, 5th ed., Sofia (2015). (in Bulgarian)

[48] A. Rahnev, K. Garov, O. Gavrailov, *Textbook for extracurricular work using BASIC*, MNP Press, Sofia (1985). (in Bulgarian)

[49] A. Rahnev, K. Garov, O. Gavrailov, *BASIC in examples and tasks*, Government Press "Narodna prosveta", Sofia (1990). (in Bulgarian)

[50] N. Kasakliev, *C# Programming Guide*, University Press "Paisii Hilendarski", Plovdiv (2016). (in Bulgarian)

[51] A. Rahnev, N. Pavlov, N. Valchanov, T. Terzieva, *Object Oriented Programming*, Lightning Source UK Ltd., London (2014).

[52] A. Menezes, P. Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, 5th ed., CRC Press LLC, New York (2001).

[53] D. Rachmawati, M. Budiman, On Using The First Variant of Dependent RSA Encryption Scheme to Secure Text: A Tutorial, *J. Phys.: Conf. Ser.*, (2020), 1542 012024.

[54] J. A. Erho, J. I. Consul, B. R. Japheth, Juggling Versus Three-Way-Reversal Sequence Rotation Performance Across Four Data Types, *International Journal of Scientific Research in Computer Science and Engineering*, **7** No. 6 (2019), 10–18.

[55] J. L. Butar-butar, F. Sinuhaji, Faktorisasi Polinomial Square-Free dan bukan Square-Free atas Lapangan Hingga Zp, *Jurnal Teori dan Aplikasi Matematika*, **3** No. 2 (2019), 132–142.

[56] L. Akcay, B. Ors, Comparison of RISC-V and transport triggered architectures for a post-quantum cryptography application, *Turk J Elec Eng & Comp Sci*, **29**, (2021), 321–333.

[57] Y. Fan, G. Chen, M. Cui, Formalization of Finite Field GF($2^n$) Based on COQ, *Computer Science*, **47** No. 12 (2020), 311–318.

[58] V. Strassen, Gaussian Elimination is not Optimal, *Numer. Math.* **13**, (1969), 354–356.