International Journal of Differential Equations and Applications
Volume **25**, No. 1 (2026), pages: 17-32
ISSN (Online): 1314-6084;
url: https://www.ijdea.eu



A NEW EXTENDED ERHO-OKRINYA ALGORITHM FOR CALCULATING THE GREATEST COMMON DIVISOR

Anton Iliev[1,2], Nikolay Kyurkchiev[1,2], Asen Rahnev[1,2],
Todorka Terzieva[1], Vesselin Kyurkchiev[1]
[1] Faculty of Mathematics and Informatics,
University of Plovdiv Paisii Hilendarski,
24, Tzar Asen Str., 4000 Plovdiv, Bulgaria
[2] Centre of Excellence in Informatics and
Information and Communication Technologies,
1113 Sofia, Bulgaria

ABSTRACT: In this paper, we present the new extended algorithm being generalization of "Algorithm 3" from [1] and slightly optimized in [2]. This algorithm can be used instead of the usual realizations of the extended Euclidean algorithm [34]. Computational process presented here is suitable for numbers of so-called normal length.

AMS Subject Classification: 11A05, 68W01
Key Words: extended Euclidean algorithm, greatest common divisor, extended Erho-Okrinya algorithm, extended Harris algorithm, computational process design

Received: February 05, 2026 **Revised:** March 10, 2026
Published: May 14, 2026 **doi:** 10.12732/ijdea.v25i1.2
Math Team /MT/ – 1991, Ltd. https://www.acadpubl.eu/ap/

1. INTRODUCTION

Let a and b be arbitrary natural numbers. We will organize new extended algorithm for finding such integer numbers x and y for which $x*a+y*b =$ Greatest Common Divisor (gcd) of a and b. The mathematical basis and theory of greatest common divisor is well present in many papers and books [3]–[38]. New implementations of these applied algorithms can be studied in [39]–[54].

For testing purposes we will use the following PC configuration:

- Processor – Intel(R) Core(TM) i7-8565U CPU 1.80GHz, 1992 Mhz, 4 Core(s), 8 Logical Processor(s);
- Installed Physical Memory (RAM) – 8.00 GB;
- System Manufacturer – Acer;
- System Model – Swift SF314-55G;
- System Type – x64-based PC;
- OS Name – Microsoft Windows 11 Enterprise x64;
- Software Environment – Microsoft Visual Studio Community 2026 x64.

In our previous research we developed [24] extended algorithm using least absolute remainder technique but here we will use natural extended Harris algorithm for testing purposes adapted from the same paper [24]. The reason of this choice is because the Harris algorithm in [24] is quite optimized:

Algorithm 1.

```
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
int g = 0;
if ((a & 1) == 0 && (b & 1) == 0)
    do { a >>= 1; b >>= 1; g++; }
    while ((a & 1) == 0 && (b & 1) == 0);
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; }
}
if (u > v) do
{
    q = u / v; u %= v;
    if (u < 1) { x = y1; y = y2; gcd = v << g; break; }
    x1 -= q * y1; x2 -= q * y2;
    if ((u & 1) == 0)
    {
        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);
        if (u == 1) { x = x1; y = x2; gcd = u << g; break; }
    }
    else
    {
        u = v - u; x1 = y1 - x1; x2 = y2 - x2;
        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);
        if (u == 1) { x = x1; y = x2; gcd = u << g; break; }
    }
    q = v / u; v %= u;
    if (v < 1) { x = x1; y = x2; gcd = u << g; break; }
    y1 -= q * x1; y2 -= q * x2;
    if ((v & 1) == 0)
    {
```

```
        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) { x = y1; y = y2; gcd = v << g; break; }
    }
    else
    {
        v = u - v; y1 = x1 - y1; y2 = x2 - y2;
        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) { x = y1; y = y2; gcd = v << g; break; }
    }
}
while (true);
else do
{
    q = v / u; v %= u;
    if (v < 1) { x = x1; y = x2; gcd = u << g; break; }
    y1 -= q * x1; y2 -= q * x2;
    if ((v & 1) == 0)
    {
        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) { x = y1; y = y2; gcd = v << g; break; }
}
else
{
    v = u - v; y1 = x1 - y1; y2 = x2 - y2;
    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) { x = y1; y = y2; gcd = v << g; break; }
}
q = u / v; u %= v;
if (u < 1) { x = y1; y = y2; gcd = v << g; break; }
x1 -= q * y1; x2 -= q * y2;
if ((u & 1) == 0)
{
    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);
     if (u == 1) { x = x1; y = x2; gcd = u << g; break; }
}
else
{
    u = v - u; x1 = y1 - x1; x2 = y2 - x2;
    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);
        if (u == 1) { x = x1; y = x2; gcd = u << g; break; }
        }
}
while (true);
```

as well as its recursive implementation as

Algorithm 2.

```
static long Extended_Harris_rec(long u, long v, long a, long b,
ref long x, ref long y, long x1, long x2, long y1, long y2, int g)
{
    long q;
    if (u > v)
    {
        q = u / v; u %= v;
        if (u < 1) { x = y1; y = y2; return v << g; }
        x1 -= q * y1; x2 -= q * y2;
        if ((u & 1) == 0)
        {
           if ((x1 & 1) == 0 && (x2 & 1) == 0)
           { x1 >>= 1; x2 >>= 1; }
           else { x1 = (x1 + b) >> 1; x2 = (x2 - a) >> 1; }
           return
           Extended_Harris_rec(u >> 1, v, a, b, ref x,
           ref y, x1, x2, y1, y2, g);
        }
        else
        {
           if (u == 1) { x = x1; y = x2; return u << g; }
           u = v - u; x1 = y1 - x1; x2 = y2 - x2;
           if ((u & 1) == 0)
           {
             if ((x1 & 1) == 0 && (x2 & 1) == 0)
             { x1 >>= 1; x2 >>= 1; }
```

```
            else { x1 = (x1 + b) >> 1; x2 = (x2 - a) >> 1; }
            return
            Extended_Harris_rec(u >> 1, v, a, b, ref x,
            ref y, x1, x2, y1, y2, g);
         }
      }
   }
   else
   {
      q = v / u; v %= u;
      if (v < 1) { x = x1; y = x2; return u << g; }
      y1 -= q * x1; y2 -= q * x2;
      if ((v & 1) == 0)
      {
         if ((y1 & 1) == 0 && (y2 & 1) == 0)
         { y1 >>= 1; y2 >>= 1; }
         else { y1 = (y1 + b) >> 1; y2 = (y2 - a) >> 1; }
         return
         Extended_Harris_rec(u, v >> 1, a, b, ref x,
         ref y, x1, x2, y1, y2, g);
      }
      else
      {
         if (v == 1) { x = y1; y = y2; return v << g; }
         v = u - v; y1 = x1 - y1; y2 = x2 - y2;
         if ((v & 1) == 0)
         {
            if ((y1 & 1) == 0 && (y2 & 1) == 0)
            { y1 >>= 1; y2 >>= 1; }
            else { y1 = (y1 + b) >> 1; y2 = (y2 - a) >> 1; }
            return
            Extended_Harris_rec(u, v >> 1, a, b, ref x,
            ref y, x1, x2, y1, y2, g);
         }
      }
   }
```

```
    return Extended_Harris_rec(u, v, a, b, ref x,
                    ref y, x1, x2, y1, y2, g);
}
```

2. MAIN RESULTS

We propose the following new Extended Algorithms 3. and 4.:

Algorithm 3.

```
if (a > b)
{
    x1 = 1; x2 = 0;
    y1 = 0; y2 = 1;
    D = (a - 1) % b + 1;
    do
    {
        r = (a - D) / b;
        x1 -= r * y1;
        x2 -= r * y2;
        b2 = (b - 1) % D + 1;
        r = (b - b2) / D;
        y1 -= r * x1;
        y2 -= r * x2;
        x1 += y1;
        x2 += y2;
        b = b2;
        a = b + D;
        D = (a - 1) % b + 1;
    } while (b != D);
    gcd = b;
    x = y1;
    y = y2;
}
else
{
```

```
    x1 = 0; x2 = 1;
    y1 = 1; y2 = 0;
    D = (b - 1) % a + 1;
    do
    {
        r = (b - D) / a;
        x1 -= r * y1;
        x2 -= r * y2;
        a2 = (a - 1) % D + 1;
        r = (a - a2) / D;
        y1 -= r * x1;
        y2 -= r * x2;
        x1 += y1;
        x2 += y2;
        a = a2;
        b = a + D;
        D = (b - 1) % a + 1;
    } while (a != D);
    gcd = a;
    x = y1;
    y = y2;
}
```

as well as its recursive version as

Algorithm 4.

```
static long Extended_Erho_Okrinya_Alg_3_rec(long a, long b,
ref long x, ref long y, long x1, long x2, long y1, long y2)
{
    long D, r, b2;
    D = (a - 1) % b + 1;
    if (b == D)
    {
        x = y1; y = y2; return b;
    }
    else
    {
```

```
        r = (a - D) / b;
        x1 -= r * y1;
        x2 -= r * y2;
        b2 = (b - 1) % D + 1;
        r = (b - b2) / D;
        y1 -= r * x1;
        y2 -= r * x2;
        x1 += y1;
        x2 += y2;
        b = b2;
        a = b + D;
        return
        Extended_Erho_Okrinya_Alg_3_rec(a, b, ref x,
        ref y, x1, x2, y1, y2);
    }
}
```

The recursive Algorithms 2. and 4. can be summoned by:

```
int g = 0;
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
if ((a & 1) == 0 && (b & 1) == 0)
    do { a >>= 1; b >>= 1; g++; }
    while ((a & 1) == 0 && (b & 1) == 0);
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; }
```

```
}
gcd = Extended_Harris_rec(u, v, a, b, ref x,
ref y, x1, x2, y1, y2, g);
```

and

```
x1 = 1; x2 = 0; y1 = 0; y2 = 1;
if (a > b)
    gcd = Extended_Erho_Okrinya_Alg_3_rec(a, b, ref x,
          ref y, x1, x2, y1, y2);
else
    gcd = Extended_Erho_Okrinya_Alg_3_rec(b, a, ref x,
          ref y, y1, y2, x1, x2);
```

respectively.

3. NUMERICAL EXAMPLE

We will examine the described above new Algorithms 3. and 4. as well as adapted Algorithms 1. and 2. from [24] with the next example:

```
long a, b, d = 0, r, gcd = 0;
long x1 = 0, x2 = 0, y1 = 0, y2 = 0;
long D, b2, a2, x = 0, y = 0, q, u, v;
for (int i = 1; i < 100000001; i++)
{
    a = i; b = 200000002 - i;
    //here are placed the source code of algorithms 1 and 3
    //as well as summoning of recursive algorithms 2 and 4.
    d += gcd;
}
Console.WriteLine(d);
```

CPU time of Algorithm 1 is: **31.809 seconds.**

CPU time of Algorithm 2 is: **41.416 seconds.**

CPU time of Algorithm 3 is: **27.601 seconds.**

CPU time of Algorithm 4 is: **33.210 seconds.**

4. CONCLUSION

The results of the numerical experiment give us reason that the new Algorithms 3. and 4. are correct and can be successfully applied in practice. In particular Algorithms 3. and 4. give better computational results in comparison to Algorithms 1. and 2. respectively.

ACKNOWLEDGMENT

This study is supported by the Centre of Excellence in Informatics and ICT under the Grant No BG16RFPR002-1.014-0018-C01, financed by the Research, Innovation and Digitalization for Smart Transformation Programme 2021–2027 and co-financed by the European Union.

REFERENCES

[1] J. Erho, A. Okrinya, Novel Approach to Shortening Convergence of Euclidean's Greatest Common Divisor Algorithm, *Mathematics Open*, (2026), **05**, 2550022

[2] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, P. Kyurkchiev, A More Efficient Description of the Erho-Okrinya Algorithm for Calculating the Greatest Common Divisor, *International Electronic Journal of Pure and Applied Mathematics*, (2026), 20, No. 1, (in print)

[3] W. Su, X. Wu, Y. Zhao, NL2ACSL: Interactively Translating Natural Language to ANSI C Specification Language with Large Language Models, *Knowledge-Based Systems*, (2025)

[4] 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)

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

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

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

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

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

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

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

[12] 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*, (2018), 120, No. 3, 379–388

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

[14] 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*, (2018), **118**, 31–37

[15] 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*, (2018), **118**, 713–721

[16] 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*, (2018), **4**, No. 3, 26–29

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

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

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

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

[21] 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, 52–58, (in Bulgarian)

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

[23] A. Iliev, N. Kyurkchiev, A. Rahnev, V. Kyurkchiev, New Extended Based on Generalization of Harris Algorithm, *Communications in Applied Analysis*, (2022), **26**, No. 1, 61–73

[24] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, A Note on Extended Based On Generalization of Harris Algorithm, *International Electronic Journal of Pure and Applied Mathematics*, (2023), 17, No. 1, 53–64

[25] A. Iliev, N. Kyurkchiev, A. Rahnev, A Refinement of the Extended Euclidean Algorithm, (2021), *International Electronic Journal of Pure and Applied Mathematics*, (2021), 15, No. 1, 33–44

[26] A. Iliev, N. Kyurkchiev, A. Rahnev, S. Enkov, An Effective Realization of Daykin-Harris Algorithm, *International Journal of Differential Equations and Applications*, (2024), 23, No. 1, 29–40

[27] A. Iliev, N. Kyurkchiev, A. Rahnev, S. Enkov, Extended Based On Generalized Daykin-Harris Algorithm, *International Journal of Differential Equations and Applications*, (2024), 23, No. 1, 41–49

[28] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, Extended Based on Generalized Tembhurne-Sathe Algorithm Using Sgn Function, *International Journal of Differential Equations and Applications*, (2023), 22, No. 1, 47–56

[29] 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*, (2018), 118, No. 2, 281–290

[30] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, New Extended Based on Generalization of Tembhurne-Sathe Algorithm, *International Journal of Differential Equations and Applications*, (2022), 21, No. 2, 89–100

[31] A. Iliev, N. Kyurkchiev, A. Rahnev, T. Terzieva, Efficient Binary Extended Algorithm Using SGN Function, *International Journal of Differential Equations and Applications*, (2021), 20, No. 2, 179–186

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

[33] A. Iliev, N. Kyurkchiev, A. Rahnev, New Extended Algorithm Using Least Absolute Remainder, *International Journal of Differential Equations and Applications*, (2022), 21, No. 1, 85–92

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

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

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

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

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

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

[40] C. Falcon Rodriguez, M. Cruz, C. Falcon, Full Euclidean Algorithm by Means of a Steady Walk, *Applied Mathematics*, (2021), **12**, 269–279

[41] V. Matanski, An Efficient Binary Algorithm for Solving Equation $GCD * 2^{|J-K|} = X * A0 + Y * B0$, *Proc. of Anniversary International Scientific Conference "Computer Technologies and Applications", 15-17 September 2021, Pamporovo, Bulgaria*, Plovdiv University Press, 79–86, ISBN: 978-619-202-702-5

[42] H. Gyulyustan, A Note on Euclidean Sequencing Algorithm, *Proc. of the Scientific Conference "Innovative ICT for Digital Research Space in Mathematics, Informatics and Educational Pedagogy"*, Pamporovo, 7-8.11.2019, Plovdiv University Press, (2020), 57–63, ISBN: 978-619-202-572-4

[43] P. Kyurkchiev, V. Matanski, The Faster Euclidean Algorithm for Computing Polynomial Multiplicative Inverse, *Proc. of the Scientific Conference "Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies"*, Pamporovo, 29-30 November 2018, (2019), 43–48, ISBN: 978-619-202-439-0

[44] V. Matanski, P. Kyurkchiev, The Faster Lehmer's Greatest Common Divisor Algorithm, *Proc. of the Scientific Conference "Innovative ICT in Research and Education: Mathematics, Informatics and Information Technologies"*, Pamporovo, 29-30 November 2018, (2019), 37–42, ISBN: 978-619-202-439-0

[45] Z. Ashraf, A. Sohail, M. Iqbal, Design and Performance Evaluation of an Authentic End-to-End Communication Model on Large-Scale Hybrid IPv4-IPv6 Virtual Networks to Detect MITM Attacks, *Cryptography*, (2024), **8**, No. 4, 49

[46] L. Babenko, V. Starodubcev, Estimation of The Execution Time of Encryption, Decryption, and Homomorphic Calculations Using The Domingo-Ferrer Cryptosystem, *Izvestiya SFedU. Engineering Sciences*, (2024), No 5, 6–15

[47] A. Ramamoorthy, A. Dutta, M. Lakshmi, Enhanced Encryption of Lunar Hyperspectral Data using Blinded Multi Prime RSA Cryptosystem, *J Indian Soc Remote Sens*, (2025), 16 pp.

[48] H. Li, L. Deng, Y. Gu, N. Wang, Y. Chen, S. Li, A secure data sharing scheme based on searchable public key encryption for authorized multi-receiver, *Journal of Systems Architecture*, (2025), **167**, 103489

[49] K. Song, J. Wei, H. Hu, X. Hou, L. Yan, S. Zang, Design and Implementation of a Parallel Stein Algorithm Based on a Ternary Optical Computer, *Appl. Opt.*, (2025), **64**, 6888–6896

[50] H. Zhang, H. Zhang, C. Xia, Z. Huang, Y. Chen, A. Barnard, ROCKET: An RNS-Based Photonic Accelerator for High-Precision and Energy-Efficient DNN Training, *Proc. of the 2025 International Conference on Supercomputing (ICS'25)*, Salt Lake City, UT, USA, 8–11 June 2025, 14 pages

[51] L. Babenko, V. Starodubcev, N. Yelchaninova, Estimation of the Search Time for Key Components in a Known Plaintext Attack on the Domingo-Ferrer Cryptosystem, *Izvestiya SFedU. Engineering Sciences*, (2025), No. 3, 110–118

[52] Z. Ibran, E. Aljatlawi, A. Awin, On Continued Fractions and Their Applications, *Journal of Applied Mathematics and Physics*, (2022), **10**, 142–159

[53] N. Boubaker, Comprehensive Insights into Matrix Continued Fractions and Their Uses, *North African Journal of Scientific Publishing*, (2025), **3**, 53–61

[54] P. Dumka, R. Chauhan, T. Verma, D. Mishra, Understanding Kuttaka (Pulverizer) method by Bhaskara II with the help of Microsoft excel, *Journal of Mathematical Problems, Equations and Statistics*, (2025), **6**, No. 1, 28–31