GPU Implementation of Quantum Secure ABC Cryptosystem on Cuda

Sedat Akleylek[a], Ramazan Koyutürk[b] and Hakan Kutucu[c]

[a] *Ondokuz Mayıs University, Department of Computer Engineering, Samsun, Turkey*
[b] *Ege University, Department of Mathematics, Izmir, Turkey*
[c] *Karabuk University, Department of Computer Engineering, Karabuk, Turkey*

Abstract

In this paper, we consider the ABC cryptosystem based on multivariate polynomial systems which is one of the post-quantum cryptosystems. We review the theoretical structure of the ABC cryptosystem and implement it on the GPU by using the NVIDIA CUDA technology. We carry out the GPU and CPU implementation details of the ABC cryptosystem on three computers with different graphics cards. We also give a comprehensive comparison between the implementations. We compute the required number of arithmetic operations for each process: key generation, encryption and decryption. According to the experimental results, the GPU implementations have better memory performance than the CPU implementations. Moreover, the encryption process is faster in the GPU implementation. Due to the structure of ABC cryptosystem, the decryption process is slower in the GPU implementation.

Keywords

Post-quantum cryptography, Multivariate polynomials, GPU, CUDA

1. Introduction

Public-key encryption systems developed in the late 1970s are becoming a crucial component of communication networks. Especially with the spread of the internet, public key systems have become common with the key exchange in SSL (Secure Sockets Layer). Systems developed for the secure transmission and storage of information have enabled further study and development on public key encryption. However, it is thought that the classical public key encryption systems will lose their reliability when quantum computers working with very small cubits are developed and spread out. For this reason, it is essential to develop public key encryption systems that are resistant to quantum attacks. Besides, the reliability of the developed systems against quantum attacks should be tested.

At present, the internet and the other communication systems are mainly based on the Digital Signature Algorithm (DSA), the Elliptic Curve DSA or the Diffie-Hellman key exchange using related algorithms, RSA encryption and digital signatures [1]. These cryptosystems guarantee their reliability due to the difficulty of several theoretical problems such as integer factorization, discrete logarithm, elliptic curves, etc. However, Peter Shor showed in 1994 that each of these problems could be solved with quantum computers in polynomial time [2]. For this reason, almost all the encryption methods we use will become insecure. Multivariate public key cryptography systems are supposed to be resistant to quantum computer attacks since they rely on a multivariate polynomials over finite fields that is an NP-hard problem [3, 4].

IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine
EMAIL: sedat.akleylek@bil.omu.edu.tr (S. Akleylek); ramazankoyuturk@gmail.com (R. Koyutürk); hakankutucu@karabuk.edu.tr (H. Kutucu)
ORCID: 0000-0001-7005-6489 (S. Akleylek); 0000-0001-7105-5197 (R. Koyutürk); 0000-0001-7144-7246 (H. Kutucu)

1.1. Literature Survey

When quantum computers of the required size are available, RSA [1], Diffie-Hellman, DSA and ECC (Elliptic Curve Cryptography), which are currently the most widely used public key encryption algorithms, will become insecure. The reason for this security problem is the fact that Shor [2] algorithm, which solves the factorization and the discrete logarithm problems in polynomial time on quantum computers, can be run quickly. Therefore, alternative methods to classical encryption methods based on mathematical problems against quantum computer attacks are required. There are currently five main classes that are believed to be resistant to quantum attacks: multivariate polynomial systems, lattice, abstract, code and isogeny based cryptography. In this study, the ABC cryptosystem using multivariate polynomial systems will be considered and the details about the GPU implementation developed using the NVIDIA CUDA technology will be given.

The cryptosystems based on multivariate polynomials are becoming very fast due to the infrastructure they use, but multivariate polynomial systems generally require more memory due to the key size. This is advantageous for low-cost devices such as smart cards and RFID [5] chips. Although there are many practical multivariate signature schemes [6,7,8], the number of secure and efficient multivariate encryption schemes is low.

Multivariate polynomial systems based cryptosystems are very attractive in the post-quantum world. GeMSS, LUOV, Rainbow, MQDSS and their derivations have been proposed [8,9,10,11]. ABC is an encryption method defined for the post-quantum world. Recently, a new simple and efficient multivariate public key encryption scheme based on matrix multiplication called "simple matrix scheme" was proposed. Then, ABC derivatives whose cubic polynomials are at least randomly quadratic were given. They showed that they broke it using algebraic attacks that were as difficult as solving an equation. A generalization of the ABC scheme using a non-square matrix instead of a square matrix was defined. A new ABC version that uses tensor multiplication of matrices to eliminate decoding errors was proposed. Optimized implementation of ABC by leveraging the features of the modern x64 CPU to increase productivity was provided [12].

1.2. Motivation and Contribution

The security of the information obtained as a result of certain labor and the applications we use in our social and professional lives is vitally important. There are several encryption algorithms for the computers and for quantum computers that are currently prototypes. The ABC encryption algorithm based on multivariate polynomials is one of these algorithms. In this study, an implementation of the ABC encryption system which works on CPU and GPU has been developed to provide necessary security after quantum computers are available. We also give a comprehensive comparison between the implementations.

The organization of the paper as follows: In Section 2, we recall basic parts of the ABC cryptosystem. In Section 3, we give implementation details as well as a comprehensive comparison. We conclude the paper in Section 4.

2. ABC Cryptosystem

In this section, the details of ABC cryptosystem is given. ABC is one of the quantum attack resistant cryptosystems. In [12], an efficient implementation of ABC cryptosystem was provided.

The security of ABC cryptosystem depends on the hardness of the solution of multivariate polynomial systems. In a multivariate quadratic (MQ) polynomial system, there are m equations $(1)-(m)$ and n variables as follows:

$$p^{(1)}(x_1,\dots,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}^{(1)}\,x_ix_j+\sum_{i=1}^{n}p_i^{(1)}x_i+p_0^{(1)} \tag{1}$$

$$p^{(2)}(x_1,\dots,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}^{(2)}\,x_ix_j+\sum_{i=1}^{n}p_i^{(2)}x_i+p_0^{(2)} \qquad (2)$$

$$\vdots$$

$$p^{(m)}(x_1,\dots,x_n)=\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}^{(m)}\,x_ix_j+\sum_{i=1}^{n}p_i^{(m)}x_i+p_0^{(m)} \qquad (m)$$

In MQ problem, the aim is to find $\acute{x}=(\acute{x}_1,\dots,\acute{x}_n)$ such that $p^{(1)}(\acute{x})=0,\dots,p^{(m)}(\acute{x})=0$. It's proved that MQ problem is NP-hard when $m\cong n$ over $GF(2)$.

The encryption/decryption process of ABC cryptosystem is given in Figure 1. An invertible multivariate quadratic polynomial system is needed such as $F\colon F^n\to F^m$. Then, to hide the linear structure of public key, linear transformations are used such as $L_1\colon F^n\to F^n$ and $L_2\colon F^m\to F^m$. Then, the public key is formed as $\acute{F}=L_2oFoL_1$. The private key set includes L_1,F and L_2. ABC cryptosystem has three phases: key generation, encryption and decryption.

$$\begin{tikzcd}
d\in F^n \arrow[r, "\bar{\mathcal{F}}"] & c\in F^m \arrow[d, "L_2^{-1}"] \\
y\in F^n \arrow[u, "L_1^{-1}"] & z\in F^m \arrow[l, "\mathcal{F}^{-1}"']
\end{tikzcd}$$

Figure 1:The encryption/decryption process of ABC cryptosystem

Key Generation : Let F be a finite field with q elements. Let $s\in S$, $n=s^2$ and $m=2n$. Then generate A, B, C matrices in the following form:

$$A=\begin{pmatrix}x_1 & \cdots & x_s\\ \vdots & \ddots & \vdots\\ x_{(s-1)(s+1)} & \cdots & x_n\end{pmatrix}, B=\begin{pmatrix}b_1 & \cdots & b_s\\ \vdots & \ddots & \vdots\\ b_{(s-1)(s+1)} & \cdots & b_n\end{pmatrix}, C=\begin{pmatrix}c_1 & \cdots & c_s\\ \vdots & \ddots & \vdots\\ c_{(s-1)(s+1)} & \cdots & c_n\end{pmatrix}$$

$F[x_1,\dots,x_n]$ is obtained by using $(x_1,\dots,x_n)$ monomials. Then, by the linear combinations of $(x_1,\dots,x_n)$, $(b_1,\dots,b_n)$ and $(c_1,\dots,c_n)$ are computed. $E_1=AB$ and $E_2=AC$ are calculated.
Two invertible linear maps $L_2\colon F^m\to F^m$ and $L_1\colon F^n\to F^n$ are randomly chosen. Then, the public key is the composition $\acute{F}=L_2oFoL_1\colon F^n\to F^m$. The private key includes B, C matrices and L_1, L_2 linear transformations.

Encryption: Let $d\in F^n$ be a message (plaintext). Then, the ciphertext is computed as follows: $c=\acute{F}(d)$ and $c\in F^m$.

Decryption: After receiving the ciphertext, $z=L_2^{-1}(c), y=F^{-1}(z)$ and $d=L_1^{-1}(y)$ are performed. Then, the message $d\in F^n$ is obtained. The detailed decryption process is as follows:

1) $z=L_2^{-1}(c)$ is computed. Then, $z\in F^n$ is used to form $\acute{E}_1$ and $\acute{E}_2$ matrices.

$$E_1=\begin{pmatrix}z_1 & \cdots & z_s\\ \vdots & \ddots & \vdots\\ z_{(s-1)(s+1)} & \cdots & z_n\end{pmatrix}, E_2=\begin{pmatrix}z_{n+1} & \cdots & z_{n+s}\\ \vdots & \ddots & \vdots\\ z_{n+(s-1)(s+1)} & \cdots & z_m\end{pmatrix}$$

2) $y = (y_1, \dots, y_n)$ is found such that $F(y) = z$.
3) $d = L_1^{-1}(y_1, \dots, y_n)$ is computed.

The decryption failure probability in Step 2 is 1/q. This means that there might be more than one solution of $y^{(1)}, \dots, y^{(l)}$. Therefore, each possible solution should be checked. In order to implement ABC cryptosystem, finite field arithmetic (polynomial addition, multiplication inversion), matrix multiplication, matrix inversion, matrix transpoze, and Gaussian elimination algorithms are needed.

3. Implementations of the ABC Algorithm

This section details the implementations of the ABC cryptosystem running on CPU and GPU separately. The ABC system operates in three main stages that are “key generation”, “encryption” and “decryption”. The parameters in the implementations on the CPU and GPU are given in Table 1. The implementations were carried out on three different platforms with the different operating systems; Windows 8 x64 and Windows 10 x64. The experiments were conducted on NVIDIA GeForce GT 740M, NVIDIA GeForce GT 840M and NVIDIA GeForce GTX 1050Ti graphics cards. One of the processes in which performance improvement is required is the addition in GF(2^8).

Table 1
Parameters in the implementations

S	8
N	64
M	128
CENTRAL_MAP_SIZE	2080
PUBLIC_KEY_SIZE	266240
SECRET_KEY_SIZE	294912

3.1. CPU Implementation

In the implementation on the CPU, after the predefined variables and the variables in the Main() function are defined, the implementation is executed by calling the functions written for key generation, encryption and decryption in the given order. The number of calls of the operations mentioned in Section 2 by three functions (key generation, encryption, decryption) is given in Table 2 for the CPU and GPU implementation.

Table 2
The number of calls of the operations in the implementation

	Key generation	Encryption	Decryption	Total
Addition	4452352	0	286720	4739072
Subtraction	0	0	2064512	2064512
Multiplication	107740992	532480	2617792	110891264
Division	0	0	16256	16256
Inverse	192	0	1	193
Matrix Inverse	2	0	0	2
Matrix Multiplication	257	0	0	257
Matrix Transposition	1	0	0	1
Computation of Coefficients	128	0	0	128
Echelon	0	0	1	1

3.2. GPU Implementation

Addition, subtraction and multiplication in the finite field GF(2^8) can be implemented on the Graphics Processing Unit (GPU) to provide performance improvement for the ABC cryptosystem.

The main task of a GPU is to display the images on the screen that are created on the computer. The first GPUs only performed this task. Over time, the Central Processing Unit (CPU) was inadequate for major computational problems, and the idea of parallel computing using the GPU is revealed. A GPGPU model has been created to provide that GPUs have a programmable interface and can be programmed in high- level languages.

Compute Unified Device Architecture (CUDA) is a parallel computing architecture introduced by NVIDIA in 2006 to take advantage of computing power of the GPU. CUDA is an application programming interface (API) model that supports programming languages such as FORTRAN, C/C++ and Python. It runs on Linux, Windows and Mac Osx platforms. Its advantages over its competitors include shared memory usage, faster data reading from the GPU, and bit-level operation.

The GPU differs from the CPU in that it has a SIMD (Single Instruction Multiple Data) architecture. Figure 2 shows the schematic comparison of CPU and GPU structures. While CPU calculations are performed in series, GPU performs calculations in parallel.

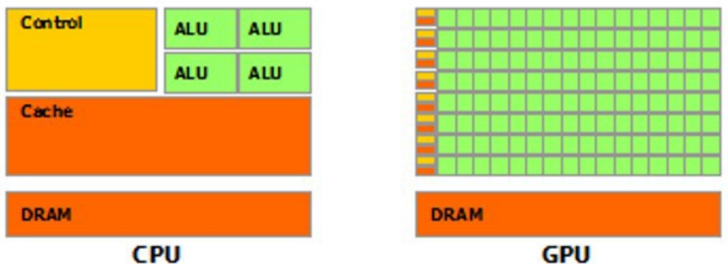

Figure 2: Comparison of CPU and CPU structures

One of the most commonly used operations in the ABC algorithm is the addition in GF(2^8). When the overall algorithm is considered, the summation function is called once in the "key generation" function and three times in the "decryption" function.

```
_ globalvoid addcuda(WORD *a, WORD *b, WORD *c) {
  *c = *a ^ *b;
}
```

Similar to the function definition in C++ programming language, but a function other than the main function is defined in CUDA. "a", "b" and "c" are pointer variables of type "WORD". "c" is the variable that will store the result of the summation of "a" and "b" and carries it over the GPU to the CPU.

Calling the "addcuda" function in the key generation function is as follows. First, "*tempA", "*tempB", "*tempC" and "tempFB" are defined as WORD data type. Then the cudaMalloc which is a pre-defined CUDA function is used to define the size of the GPU. With the cudaMemcpy function, the values on the CPU are copied to the GPU. We call the function on the GPU with "addcuda <<< 1, 1 >>> ()". After completing this process, we use the cudaMemcpy function to copy the values on the GPU back to the CPU.

```
WORD *tempA, *tempB, *tempC, tempFB;
cudaMalloc((void**)&tempA, sizeof(WORD));
cudaMalloc((void **)&tempB, sizeof(WORD));
cudaMalloc((void **)&tempC, sizeof(WORD));
for (s = 0; s < VARIABLE; s++) {
    for (t = s; t < VARIABLE; t++) {
        if (t == s)
            FB[flag++] = V[s * VARIABLE + t];
        else {
            cudaMemcpy(tempA, &V[s*VARIABLE+t], sizeof(WORD), cudaMemcpy HostToDevice);
            cudaMemcpy(tempB, &V[t*VARIABLE+s], sizeof(WORD), cudaMemcpy HostToDevice);
            addcuda<<<1,1>>>(tempA,tempB,tempC);
            cudaMemcpy(&tempFB,tempC,sizeof(WORD), cudaMemcpyDeviceToHost);
            FB[flag++] = tempFB;
        }
    }
}
cudaFree(tempA);
cudaFree(tempB);
cudaFree(tempC);
```

3.3. Comparison of the implementations

The implementation prepared in .Net environment using Microsoft Visual Studio for the ABC cryptosystem has been tested on three different computers, each with different graphics processors. Table 3 summarizes the computer, graphics processor, and operating system information used.

Table 3
Test environment

Computer	CPU	RAM	Graphics Card	Operating System
1	Intel(R) Core(TM) i7-4500U CPU @ 1.80 GHz	12 GB	GeForce GT 740M 2 GB	Windows 8.1
2	Intel(R) Core(TM) i7-4210U CPU @ 1.70 GHz	8 GB	GeForce GT 840M 2 GB	Windows 10
3	Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz	16 GB	GeForce GTX 1050Ti 4 GB	Windows 10

In Table 4, the total running time of both CPU and GPU implementations prepared for the ABC cryptosystem in three different environments are given in microseconds (µs).

Table 4
Execution times of the implementations

	CPU	GPU
Computer1	4336000 µs	6089299 µs
Computer2	5123000 µs	6589299 µs
Computer3	2214000 µs	4856900 µs

Table 5 shows the memory usage of the implementations prepared for the ABC cryptosystem on the CPU for three different environments.

Table 5
Memory usage of the implementations

	CPU	GPU
Computer1	90 MB	76 MB
Computer2	100 MB	87 MB
Computer3	127 MB	118 MB

In Table 6, the running times of the encryption and decryption functions on both CPU and GPU implementations prepared for the ABC cryptosystem in three different environments are given in microseconds (μs).

Table 6
Execution times of the encryption and decryption functions

	CPU		GPU	
	Encryption	Decryption	Encryption	Decryption
Computer1	16000 μs	132000 μs	8000 μs	3291599 μs
Computer2	19000 μs	145000 μs	11000 μs	3701789 μs
Computer3	10000 μs	78000 μs	3000 μs	2479900 μs

The running times of the implementations and the functions required for the ABC cryptosystem shown in Table 2 in detail are evaluated by counting the number of times the key generation, encryption and decryption are called and used. As can be noticed in the call of functions the entire addition operations in GF(2^8) are performed on the GPU. Furthermore, although an increase in total time has been observed, an improvement has been achieved in the process of encryption. In addition to running time improvements, there is an improvement in memory usage. The running time in Table 4 is measured for both implementations as follows: time starts before the key generation and ends with the completion of the decryption.

The running time of the CPU implementation is 4336000 microseconds in Computer1. The running time of GPU implementation is 6089299 microseconds on the same computer. There is a 40% increase in time compared to the CPU implementation. However, the memory consumption of the CPU implementation is 90MB while the memory consumption of the GPU implementation is 76MB. That is, a 15% memory saving is achieved compared to the CPU implementation. In the encryption function, the time is reduced from 16000 microseconds to 8000 microseconds and a 50% time-saving is achieved.

The running time of the CPU implementation is 5123000 microseconds in Computer2. The running time of GPU implementation is 6589299 microseconds on the same computer. There is a 28% increase in time compared to the CPU implementation. However, the memory consumption of the CPU implementation is 100MB while the memory consumption of the GPU implementation is 87MB. That is, a 13% memory saving is achieved compared to the CPU implementation. In the encryption function, the time is reduced from 19000 microseconds to 11000 microseconds and a 42% time-saving is achieved.

The running time of the CPU implementation is 2214000 microseconds in Computer3. The running time of GPU implementation is 4856900 microseconds on the same computer. There is a 19% increase in time compared to the CPU implementation. However, the memory consumption of the CPU implementation is 127MB while the memory consumption of the GPU implementation is 118MB. That is, a 7% memory saving is achieved compared to the CPU implementation. In the encryption function, the time is reduced from 10000 microseconds to 3000 microseconds and a 70% time-saving is achieved.

Regardless of the working environment, the GPU implementation is slower than the CPU implementation, but it is better in memory consumption. The reason for this difference is that the addition operations in GF(2^8) are performed on the GPU.

4. Conclusion

It is believed that the ABC, one of the multivariate public key cryptosystems, will be robust to computational attacks using quantum computers. As a result of this study, an application or implementation that works with a graphics processor regardless of the working environment of the computers is better in memory usage. The running times are evaluated by counting the number of calls of the key generation, encryption and decryption. The number of the arithmetic operations is computed. The memory consumption of the CPU implementation is larger than the GPU implementation in all platforms. In addition to the efficiency in terms of memory consumption, running the entire implementation on the GPU will allow better results to increase performance in terms of time.

Acknowledgement

The first and second authors were partially supported by TÜBİTAK under grant no. EEEAG-116E279.

5. References

[1] R. L. Rivest, A. Shamir, L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. Commun, ACM 21 2 (1978) 120–126

[2] P. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Computing 26 5 (1997) 1484–1509

[3] J. A. Buchmann, D. Butin, Post-Quantum Cryptography: State of the Art. The New Code-breakers, LNCS 9100 (2016) 88-108.

[4] N. Kundu, S.M. Debnath, D. Mishra, A Secure and Efficient Group Signature Scheme Based On Multivariate Public Key Cryptography, Journal of Information Security and Applications 58 102776 (2021). doi: 10.1016/j.jisa.2021.102776

[5] J. Chen, J. Ning, J. Ling, T.S.C. Lau, Y. Wang, A New Encryption Scheme For Multivariate Quadratic Systems, Theoretical Computer Science 809 (2020) 372-383

[6] N. Kundu, S.M. Debnath, D. Mishra, T. Choudhury, Post-Quantum Digital Signature Scheme Based On Multivariate Cubic Problem, Journal of Information Security Applications, 53 (2020) 102512. doi: 10.1016/j.jisa.2020.102512

[7] D. Smith-Tone, C. Tone, A Multivariate Cryptosystem Inspired By Random Linear Codes, Finite Fields and Their Applications 69 101778 (2021). doi: 10.1016/j.ffa.2020.101778

[8] Post-Quantum Cryptography Round 2 Submissions, 2020. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

[9] M. S. Chen, A. Hülsing, J. Rijneveld, S. Samardjiska, P. Schwabe, From 5-pass MQ-based identification to MQ-based signatures, in: Proceedings of 22th Annual International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT, Hanoi, 2016, vol. 10032, pp. 135-165

[10] S. Akleylek, M. Soysaldı, A novel 3-pass identification scheme and signature scheme based on multivariate quadratic polynomials, Turkish Journal of Mathematics 43 1 (2019) 241-257

[11] S. Akleylek, M. Soysald, W.-K. Lee, S.O. Hwang, D. Chee-Keong Wong,Novel Post-quantum MQ-based Signature Scheme for Internet of Things with Parallel Implementation, IEEE Internet of Things Journal (2021). doi:10.1109/JIOT.2020.3038388, 2021.

[12] Z. Peng, S. Tang, J. Chen, C. Wu and X. Zhang, Fast Implementation of Simple Matrix Encryption Scheme on Modern x64 CPU, in: ISPEC 2016, 10060, pp. 151-166.