Asesorías y Tutorías para la Investigación Científica en la Educación Puig-Salabarría S.C.
José María Pino Suárez 400-2 esq a Lerdo de Tejada. Toluca, Estado de México. 7223898475
RFC: ATI120618V12

Revista Dilemas Contemporáneos: Educación, Política y Valores.
http://www.dilemascontemporaneoseducacionpoliticayvalores.com/
Año: VI Número: Edición Especial Artículo no.:75 Período: Diciembre 2018.

TÍTULO: Investigación y optimización del Sistema de Cifrado ElGamal.

AUTOR:

1. Alisher R. Zhumaniezov.

RESUMEN: Este artículo describe la optimización del esquema de encriptación ElGamal mediante la introducción de diferentes esquemas de optimización. Entre todos los esquemas, se eligieron los siguientes: reducción de Barrett, reducción de Montgomery y reducción de Montgomery con esquemas de optimización de almacenamiento en caché. El algoritmo de cifrado de clave pública El Gamal fue propuesto por Taher Elgamal en 1985. El esquema completo fue desarrollado en base al protocolo Diffie-Hellman. La estabilidad criptográfica de este algoritmo se basa en la dificultad de calcular el logaritmo discreto en un campo finito. La falta de una patente para un algoritmo hace que sea una alternativa más barata al algoritmo RSA.

PALABRAS CLAVES: esquema de encriptación ElGamal, módulo de exponenciación binaria, reducción de Barrett, reducción de Montgomery, aritmética de bits.

TITLE: Research and optimization of ElGamal Encryption System.

AUTHOR:

1. Alisher R. Zhumaniezov.

ABSTRACT: This article describes the optimization of ElGamal encryption scheme through the introduction of different optimization schemes. Among all the schemes, the following were chosen: Barrett reduction, Montgomery reduction and Montgomery reduction with caching optimization schemes. ElGamal public key encryption algorithm was proposed by Taher Elgamal in 1985. The complete scheme was developed based on the Diffie-Hellman protocol. The cryptographic stability of this algorithm is based on the difficulty of calculating the discrete logarithm in a finite field. The lack of a patent for an algorithm makes it a cheaper alternative to the RSA algorithm.

KEY WORDS: ElGamal encryption scheme, Binary exponentiation modulo, Barrett's reduction, Montgomery reduction, bit arithmetic.

INTRODUCTION.

Cryptography is a science engaged in the development of cryptosystems, that is, the study of mathematical methods for transforming information in order to protect systems from unauthorized access.

The goal of this paper is to optimize the implementation of ElGamal encryption system. Our main goal is to reduce time for encryption and decryption. To do this, we will optimize the work of exponentiation. We must also measure winnings by time.

In the age of modern technology, time is a very valuable resource. Therefore, an important indicator of the implementation is its calculation speed. The optimization process goes in two directions:

1. Reducing the asymptotics of the algorithm. Extremely difficult task, as it is not always possible, but it guarantees the gain in time for large values of the parameters.

2. The decrease in the coefficient in the asymptotics. There are several ways to achieve: parallelization, reduction to bit arithmetic, splitting into subtasks, etc. The main difficulty is that it is sometimes difficult to calculate the resulting change in a constant.

DEVELOPMENT.

The ElGamal public-key encryption algorithm was proposed by Taher Elgamal in 1985 [Elgamal T. A., 1985]. The whole scheme was developed based on the Diffie-Hellman protocol. The cryptographic stability of this algorithm is based on the difficulty of calculating the discrete logarithm in a finite field. The lack of a patent for an algorithm makes it a cheaper alternative to the RSA algorithm.

The most complex operations in exponentiation are modulo operations and multiplication operations. Therefore, our work will be aimed at optimizing precisely these operations.

In the course of work, we will look at the effectiveness of well-known optimizations. We will choose the most effective ones and on the basis of it we will implement an optimized cryptosystem.

Methods.

ElGamal encryption scheme.

The encryption algorithm consists of three stages [Б. А. Фороузан; Menezes A. J. et.al., 1996]:

1. Key generation:

a. A random prime p is chosen.

b. The number g is chosen - the primitive root of p.

c. A random number x is selected in the interval $[2, p-1]$

d. Calculated $y = g^x \ mod \ p$.

e. (p, g, y) - is public key. x is the private key.

2. Encryption Algorithm:

a. The message is divided into blocks.

b. Each block is encoded by the number m_i.

c. A session key k is selected in the interval $[2, p-2]$.

d. The numbers $a_i = g^k \ mod \ p$; $b_i = y^k * m_i \ mod \ p$.

e. A pair of numbers (a_i, b_i) is called the ciphertext of the i block.

3. Decryption Algorithm:

a. The value of the block is restored by the formula $m_i = b_i * a_i^{-x} \ mod \ p$.

b. According to the values of the block, we restore the original message.

Let us prove the correctness of the algorithm.

1. $y = g^x \ mod \ p \Rightarrow b_i = y^k * m_i \ mod \ p = g^{kx} * m_i \ mod \ p$

2. $a_i^{-x} \ mod \ p = g^{-kx} \ mod \ p$

3. $b_i * a_i^{-x} \ mod \ p = (g^{kx} * m_i \ mod \ p) * (g^{-kx} \ mod \ p) = (g^{kx} * m_i * g^{-kx}) mod \ p = m_i$

Thus, the correctness of the algorithm is proved.

Binary exponentiation modulo.

Exponentiation modulo using of sequential multiplication by a number n times is too long, then in practice a binary exponentiation algorithm is used.

This algorithm is based on the following idea [Рябко Б. Я., et. al., 2004]:

1. $a^{2*k} \ mod \ p = (a^k \ mod \ p)^2 \ mod \ p$

2. $a^{2*k+1} \ mod \ p = (a * (a^k \ mod \ p)^2) \ mod \ p$

There are two implementations of the algorithm in the direction of sorting bits of a degree: from the high-order bit to the low-order one, and vice versa, from the low-order to the highest. This paper discusses the option from the high bit to the low bit.

Algorithm Description:

1. Initially, the result is 1.

2. Enumerate all bits from high to low.

3. We square the result modulo.

4. If the current bit is one, then multiply the result by the original number a.

5. Go to the next bit.

Barrett's reduction.

Another option for replacing the taking modulo is Barrett's reduction.

The main idea of Barrett's reduction is the representation of the remainder in the following form [Cao, Z. et. al., 2014; Lim C. H, et. al., 1997]:

$$x \bmod n = x - \left\lfloor \frac{x}{n} \right\rfloor * n \quad (1)$$

$$\left\lfloor \frac{x}{n} \right\rfloor = \left\lfloor \frac{x*4^k}{n*4^k} \right\rfloor \approx \left\lfloor \frac{x*m}{4^k} \right\rfloor, \text{ where } m = \left\lfloor \frac{4^k}{n} \right\rfloor \quad (2)$$

For greater accuracy and speed of operation, take the minimum k, such that $2^k > n$, that is, $k = \lceil log_2 n \rceil$.

Thus, the Barrett reduction is calculated as follows [5] [6]:

$$q = \left\lfloor \frac{x*m}{4^k} \right\rfloor \quad (3)$$

$$u = x - q * n \quad (4)$$

$$x \bmod n = \begin{cases} u, if\ u < n \\ u - n, otherwise \end{cases} \quad (5)$$

Now we give the exponential algorithm:

1. Calculate in advance $m = \left\lfloor \frac{4^k}{n} \right\rfloor$.
2. Initially, the result is 1.
3. Enumerate all bits from high to low.
4. In the result we write the reduction of Barrett from the square of the result.
5. If the current bit is equal to one, then in the result we write the Barrett's reduction from the product of the result by the number a.
6. Go to the next bit.

Montgomery Reduction.

Since the procedure for taking an arbitrary module is a time-consuming operation, they try to replace it with a taking by module - a power of two. One such method is the Montgomery reduction.

The Montgomery Theorem [Ors S. B; et.al, 2003; Белов А; 1999]:

N, R – *natural mutually simple numbers*, $N' = (-N^{-1})\ mod\ R$ => $\forall\ x\ \in\ \mathbb{N}$, $y = (x + N * ((x * N')mod\ R)) : R$, and ${}^{y}/_{R} \equiv x * R^{-1}\ mod\ N$.

We introduce the function of the Montgomery multiplication [Ors S. B., et. al. 2003, Белов, А. 1999].

$$Mont(a, b) = (a * b * R^{-1})\ mod\ N \qquad (6)$$

However, we note that the calculation is not directly effective because of the search for the inverse element, therefore, when implementing, we use the Montgomery theorem [Ors S. B; et. al., 2003; Белов, А., 1999]:

$$u = \left(a * b + N * \left((a * b * N')mod\ R\right)\right)/R \qquad (7)$$

$$Mont(a, b) = \begin{cases} u, if\ u < N \\ u - N, otherwise \end{cases} \qquad (8)$$

Note that if we use the integer power of two as R, then we can get a big time gain due to bit arithmetic.

We call the (R, N)-residual of x number:

$$\bar{x} = (x * R)\ mod\ N \qquad (9)$$

Note a number of properties required in our work [Ors S. B. et. al., 2003; Белов А., 1999]:

1. $Mont(\bar{a}, 1) = a$.
2. $Mont(\bar{a}, \bar{b}) = \overline{Mont(a, b)}$.

Now we can present the exponentiation algorithm:

1. Initially, the result is $R\ mod\ N$.
2. Calculate (R, N)-residual of a.
3. Enumerate all bits from high to low.
4. In the result, write the value of the Montgomery product of the result on the result.
5. If the current bit is equal to one, then in the result we write the Montgomery product of the result by the (R, N)-residual of a.
6. Go to the next bit.
7. Return the Montgomery product of the result and 1.

Montgomery Reduction with caching.

Also, in this paper we'll consider modification of Montgomery reduction. The main idea was using caching in (7) of multiplication $a * b$:

$$c = a * b \qquad (10)$$

$$u = \left(c + N * \left((c * N')mod\ R\right)\right)/R \qquad (11)$$

Results and Discussion.

Barrett's reduction.

In Barrett's reduction, we have next operations:

1. x: 1 multiplication.
2. q: 1 multiplication + 1 integer division by power of two => 1 multiplication + 1 shift.
3. u: 1 multiplication + 1 subtraction.
4. $x\ mod\ n$: no more than 1 subtraction.

5. Total number of operations: 3 multiplication + 1 shift + 2 subtraction

For each operation, we assume complexity [Егоров Д. Ф. 1923; Акритас А. 1994]:

$$\tau(\text{multiplication}, A) = O(\ln(A) * \ln(\ln(A))) \quad (12)$$

$$\tau(\text{bitwise and}, A) = O(\ln(A)) \quad (13)$$

$$\tau(\text{subtraction}, A) = O(\ln(A)) \quad (14)$$

Then total complexity $\tau(A)$ of one iteration was:

$$\tau(A) = 3 * \tau(\text{multiplication}, A) + \tau(\text{shift}, A) +$$

$$+2 * \tau(\text{subtraction}, A) = O(\ln(A) * \ln(\ln(A))) \quad (15)$$

Montgomery Reduction.

In Montgomery reduction we have next operations:

1. u: 4 multiplication + 1 addition + 1 integer division by power of two + 1 modulo residue by power of two => 4 multiplication + 1 addition + 1 shift + 1 bitwise and.

2. $Mont(a, b)$: no more than 1 subtraction.

3. Total number of operations: 4 multiplication + 1 addition + 1 shift + 1 bitwise and + 1 subtraction.

For each operation, we assume complexity (12), (13), (14) and [Егоров Д. Ф., 1923; Акритас А. 1994]:

$$\tau(\text{bitwise and}, A) = O(\ln(A)) \quad (16)$$

$$\tau(\text{addition}, A) = O(\ln(A)) \quad (17)$$

Then total complexity $\tau(A)$ of one iteration was:

$$\tau(A) = 4 * \tau(\text{multiplication}, A) + \tau(\text{addition}, A) + \tau(\text{shift}, A) +$$

$$+\tau(\text{bitwise and}, A) + \tau(\text{subtraction}, A) = O(\ln(A) * \ln(\ln(A))) \quad (18)$$

Montgomery Reduction with caching.

In Montgomery reduction, we have next operations:

1. x: 1 multiplication.
2. u: 2 multiplication + 1 addition + 1 integer division by power of two + 1 modulo residue by power of two => 4 multiplication + 1 addition + 1 shift + 1 bitwise and.
3. $Mont(a, b)$: no more than 1 subtraction.
4. Total number of operations: 3 multiplication + 1 addition + 1 shift + 1 bitwise and + 1 subtraction

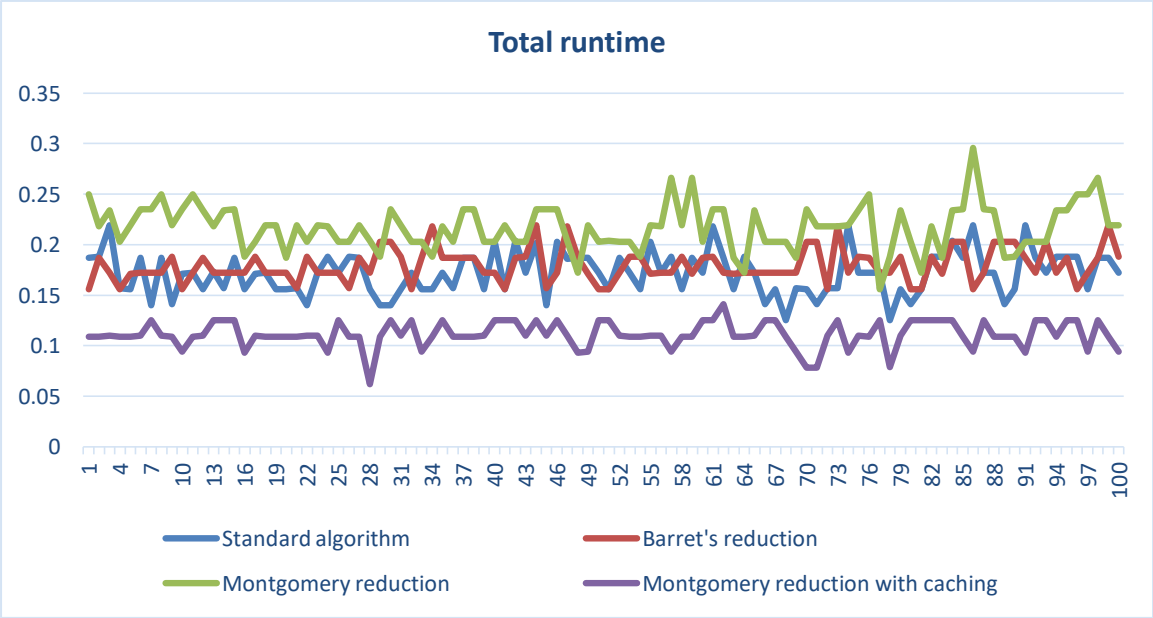
For each operation, we assume complexity (12), (13), (14), (16) and (17) [Егоров Д. Ф. 1923; Акритас А. 1994]. Then total complexity $\tau(A)$ of one iteration was:

$$\tau(A) = 3 * \tau(\text{multiplication}, A) + \tau(\text{addition}, A) + \tau(\text{shift}, A) +$$

$$+\tau(\text{bitwise and}, A) + \tau(\text{subtraction}, A) = O(\ln(A) * \ln(\ln(A))) \quad (19)$$

64-bit module.

Figure 4.1. Comparison of the total runtime with the original algorithm (64-bit).



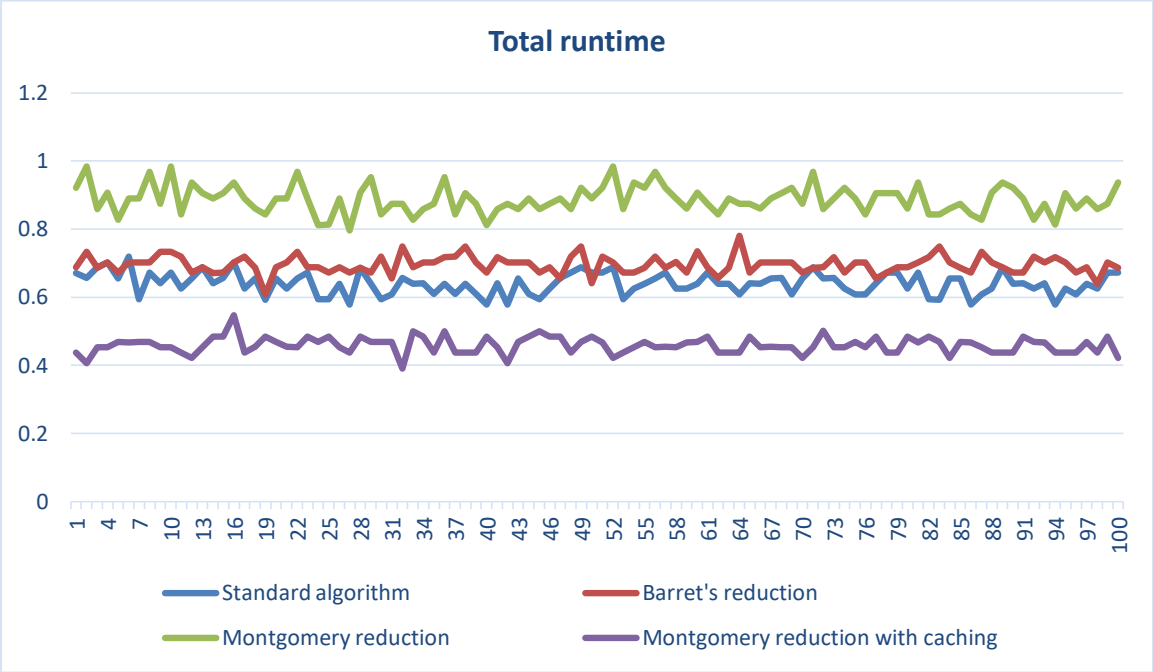
Barret’s reduction: As can be seen from the graph above, the using of Barrett's reduction hardly benefits the standard exponentiation.

Montgomery reduction: As can be seen from the graph above, the using of a pure reduction of Montgomery gives a loss in time.

Montgomery reduction with caching: As can be seen from the graph above, the using of the Montgomery reduction with caching gives a considerable gain in time.

128-bit module.

Figure 4.2. Comparison of the total runtime with the original algorithm (128-bit).



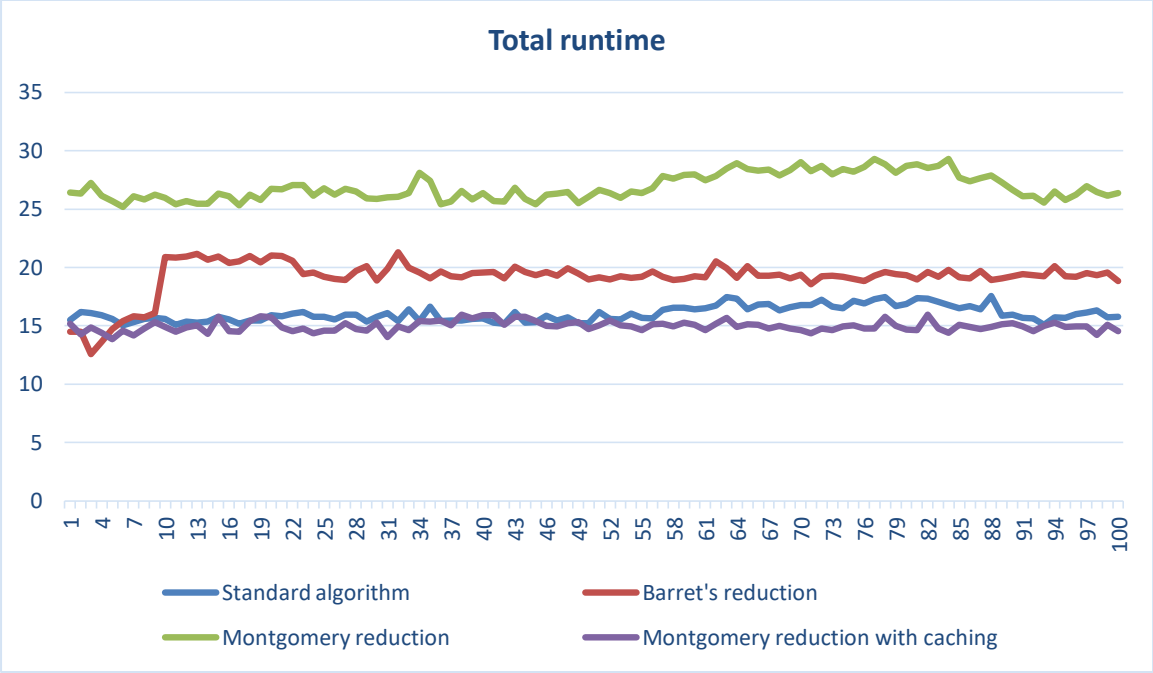
Barret’s reduction: As can be seen from the graph above, the using of Barrett's pure reduction gives a loss in time.

Montgomery reduction: As can be seen from the graph above, the using of a pure reduction of Montgomery gives a loss in time.

Montgomery reduction with caching: As can be seen from the graph above, the using of the Montgomery reduction with caching gives a considerable gain in time.

1024-bit module.

Figure 4.3. Comparison of the total runtime with the original algorithm (1024-bit).



Barret's reduction: As can be seen from the graph above, the using of Barrett's pure reduction gives a loss in time.

Montgomery reduction: As can be seen from the graph above, the using of a pure reduction of Montgomery gives a loss in time.

Montgomery reduction with caching: As can be seen from the graph above, the using of the Montgomery reduction with caching gives a considerable gain in time.

CONCLUSIONS.

The theoretical and practical questions of the ElGamal encryption system are considered, and it was developed a program containing the implementation of all the considered optimization schemes for the exponentiation modulo. Also, in the paper, to evaluate the efficiency of optimization of the exponentiation modulo, experiments were performed and the results are shown.

In the course of this work, the following conclusions were made on the schemes considered:

- Barrett's reduction — showed the result a little worse than the standard algorithm. At small values total runtime was close to standard, but at large values total runtime increased.
- Montgomery reduction — showed the worst result among all the considered schemes. Total runtime increased by more than 1.5 times.
- Montgomery reduction with caching — showed the best result among all the considered schemes. Total runtime was decreased by about 33%

The further direction of the study may be the reduction of the work time spent on one iteraction. Among the possible approaches: the disclosure of internal function calls the use of a lower-level language, and the use of a faster algorithm for selecting coefficients.

Acknowledgements.

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

BIBLIOGRAPHIC REFERENCES.

[1] Акритас А., (1994) Основы компьютерной алгебры с приложениями: пер. с англ., М., Мир, 1994, 544 с.

[2] Cao, Zhengjun; Wei, Ruizhong; Lin, Xiaodong., (2014) A Fast Modular Reduction Method. Recuperado de: https://eprint.iacr.org/2014/040.pdf

[3] Белов А., (1999) Сложность алгоритмов/ Белов А. Тихомиров В. — КВАНТ № 2, pp. 8-11

[4] Егоров Д. Ф., (1923) Элементы теории чисел., М., Петроград: Госиздат, 202 с.

[5] ElGamal T. A, (1985) Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms/ El Gamal T. — IEEE, 1985. Vol. 31, pp. 469–472.

[6] Б. А. Фороузан., (1985) Схема цифровой подписи Эль-Гамаля/ Пер. А. Н. Берлин. — Курс лекций. Recuperado de: http://www.rtsu.tj/upload/files/элподписъ7-сем.PDF

[7] Lim C. H; Hwang, H.S; Lee, P.H; Fast Modular Reduction With Precomputation. Recuperado de: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.8501

[8] Menezes A. J., Oorschot P. V., Vanstone S. A. (1996) The ElGamal signature scheme/— CRC Press, pp. 816.

[9] Ors S. B; Batina L; Preneel B; Vandewalle J; (2003) Hardware Implementation of a Montgomery Modular Multiplier in a Systolic Array/ Proceedings of the International Parallel and Distributed Processing Symposium, p. 8. DOI: 10.1109/IPDPS.2003.1213341

[10] Рябко Б. Я; Фионов А. Н., (2004) Основы современной криптографии для специалистов в информационных технологиях/ Рябко Б. Я.,— Научный мир, pp.173 с.

DATA OF THE AUTHOR.

1. Alisher R. Zhumaniezov. Kazan Federal University. Email: myzerix58@gmail.com

RECIBIDO: 4 de noviembre del 2018. **APROBADO**: 15 de noviembre del 2018.