# Efficient Computation of Parametric Exponentiation in Parametric Algebra

Khudoykulov Zarifjon[1], Khudoynazarov Umidjon[2], Muminova Mastura[3]
Department of Cryptology, Tashkent University of Information Technologies, Tashkent 100084, Uzbekistan [1]
Department of Cybersecurity & Criminology, Tashkent University of Information Technologies, Tashkent 100084, Uzbekistan [2]
Department of Pedagogy, Fergana State University, Fergana 150100, Uzbekistan [3]

***Abstract*—This study analyzes the computational efficiency of parametric exponentiation operations in parametric algebra and proposes an algorithmic approach for their fast computation. By leveraging the properties of parametric algebra, the possibility of reducing parametric exponentiation to fast exponentiation methods is established, and a corresponding computational algorithm is developed. The proposed approach enables more efficient computation of parametric exponentiation and reduces the computational impact of the system parameter. The obtained results provide a basis for reconsidering approaches to the efficiency of parametric algebra-based computations, demonstrating that the slowness of exponentiation often stems from the computational methods employed rather than the algebraic structure itself. This research establishes a significant foundation for the efficient organization of cryptographic computations based on parametric algebra and for addressing algorithmic optimization challenges.**

***Keywords*—Parametric algebra; parametric multiplication; parametric exponentiation; computational efficiency; algebraic structures; computational complexity; fast exponentiation**

## I. Introduction

The security of cryptographic algorithms depends on the computational complexity of the underlying algebraic problems. In symmetric encryption algorithms, complexity is determined by key length, structural design, and resistance to cryptanalytic attacks. In public-key cryptographic algorithms, security primarily relies on the computational complexity of problems such as discrete logarithms in finite fields, integer factorization of large numbers, and computational challenges related to elliptic curves [1, 2].

The rapid development of modern computing technologies is leading to an increase in key lengths and computational complexity to maintain security in existing cryptographic algorithms. However, this approach results in the slowing down of encryption and decryption processes and limits computational efficiency [13].

In recent years, the development of complex cryptographic computing methods based on new algebraic structures has been considered one of the most promising areas. An illustration of this phenomenon is provided by elliptic curve cryptography, which employs more intricate algebraic structures in comparison to classical modular arithmetic. Furthermore, algebraic models such as lattice structures (including LWE), error-correcting codes, and multivariate polynomial equations are actively evolving in modern cryptography. These approaches enrich the mathematical foundations of cryptographic systems and enhance the flexibility of computational processes [21, 22].

The concept of parametric algebra is predicated on the extension of classical and matrix-based algebraic structures through the introduction of an additional parameter. This algebra opens up new mathematical possibilities by generalizing classical algebraic operations through the introduction of an additional parameter. A variety of cryptographic algorithms based on parametric algebra have been proposed and are currently being studied as algebraic models with the potential for application in practical computations [3,4].

Parametric algebra is a system that is only just beginning to gain ground in cryptographic research. Since its structure differs from classical modular arithmetic and it is not yet sufficiently familiar to the wider community, the main concepts and properties are systematically presented in Section III of the study.

Research on parametric algebra has primarily focused on its theoretical properties and general algorithmic applications. However, the issues concerning the computational efficiency of parametric exponentiation operations and the possibilities of adapting them to fast exponentiation methods have not been sufficiently explored.

Existing approaches to parametric exponentiation often rely on repeated parametric multiplications, resulting in high computational overhead and limited practical efficiency. Therefore, developing efficient methods and integrating them with fast exponentiation techniques remains an important scientific problem.

The main contributions of this study can be summarized as follows:

- A novel formulation for parametric exponentiation based on modular binary exponentiation;
- Reduction of the computational dependence on the parameter $R$ in parametric exponentiation;
- Experimental validation using runtime, memory usage, and CPU tick measurements demonstrating efficiency and scalability.

This study focuses on the computational efficiency of parametric exponentiation and its algorithmic optimization.

Leveraging the properties of parametric algebra, a fast modular binary exponentiation algorithm specifically adapted for parametric structures is proposed.

## II. Related Works

### A. Parametric Algebra and Related Algebraic Approaches

Parametric algebraic operations are based on the concept of generalizing classical algebraic structures through the introduction of an additional parameter. This approach emerges independently across various fields of mathematics and cryptography. This section provides a comprehensive analysis of scientific works that are conceptually and formally related to parametric algebraic operations.

*1) Deformations of algebras and parametric operations:* In the Theory of Algebraic Deformations developed by Murray Gerstenhaber, an algebraic operation is modified through a parameter [14]. The classical multiplication operation is deformed as follows:

$$a *_t b = ab + t\varphi_1(a,b) + t^2\varphi_2(a,b) \qquad (1)$$

Here, t is the deformation parameter that generates new algebraic structures.

In parametric algebraic operations, classical operations are also modified through a parameter, but over finite fields and by arithmetic expressions. For example:

$$a®b \equiv a + b + a * R * b \ (mod \ \ n) \qquad (2)$$

The similarity between Formula (1) and Formula (2) is that, in both cases, the parameter controls the result of the operation. Therefore, parametric algebra is a discrete, cryptographically oriented variant of deformation theory.

*2) Parameter-dependent binary operations:* Recent studies show that alternative binary operations are used to construct new mathematical structures. For example, several works on triangular norms have introduced generalized spaces of non-additive measures in which the selected operation determines the resulting functional framework [16].

This illustrates a general principle: modifying or selecting a binary operation can lead to systems with new structural properties. Parametric algebra follows the same direction by introducing parameter-dependent operations over modular and finite arithmetic structures for cryptographic applications. In this sense, it can be regarded as a discrete and cryptography-oriented form of the broader theory of alternative binary operations.

*3) Partially associative algebraic structures:* In some non-associative algebraic structures, multiplication is governed by specific identities such as:

$$(xy)(xz) = x(yz)x$$

Thus, general associativity is not required, while structural properties are preserved through symmetry [17,18].

In parametric algebraic operations, associativity also depends on the chosen parameter. For certain values of \(R\), the following relation holds:

$$(a®b)®c = a®(b®)$$

This equality is satisfied only for specific parameter values.

Taken together, these studies represent key theoretical frameworks and algorithmic methodologies related to the principles of parametric algebra.

### B. Fast Exponentiation and Computational Efficiency

Efficient modular exponentiation methods have been widely studied through several classical approaches. Binary exponentiation is one of the fundamental methods and is applied in left-to-right and right-to-left forms. Its main advantage is reducing the number of multiplications from linear complexity to logarithmic complexity [24].

The Montgomery ladder improves resistance through a regular computation pattern. The sliding-window method increases efficiency by using precomputation. Techniques such as NAF, MOF, and w-NAF reduce the number of nonzero digits in the exponent representation, thereby decreasing the required multiplications [25].

In many public-key systems, the main computational cost is associated with exponentiation operations. In RSA, Diffie–Hellman, digital signature schemes, and related protocols, the exponentiation process has a significant impact on overall efficiency [15].

Recent studies have explored several approaches to address this problem. In particular, classical modular arithmetic methods such as binary, window-based, and other fast exponentiation techniques continue to be improved. In digital signature systems, special attention is also given to accelerating signature generation and verification procedures [19,20].

In specialized algebraic environments, especially Galois fields, methods for optimizing squaring and multiplication operations have been proposed. In addition, new asymptotically faster algorithms are being developed by exploiting modulus factorization or internal algebraic properties of the underlying structures [23].

These results show that improving exponentiation algorithms remains a theoretically and practically relevant research direction today.

Beyond classical algebraic settings, the importance of exponentiation is also significant in parametric algebra. In such systems, the main computational processes are carried out through parameter-dependent addition, multiplication, and exponentiation operations.

Table I presents a comparative summary of the main theoretical approaches related to parametric algebra and the key studies on fast exponentiation.

TABLE I. COMPARATIVE SUMMARY OF RELATED WORKS

| No | Ref | General Form of the Algebraic Operation | Algebraic Setting | Main Idea \| Relevance | Main Idea \| Relevance |
|---|---|---|---|---|---|
| 1 | Gerstenhaber [14] | Related algebraic theory | Deformation theory | Classical algebraic operations are modified through an external parameter, producing new algebraic structures. | Provides a theoretical foundation for parameter-based operations in parametric algebra. |
| 2 | Sukhorukova [16] | Parameter-dependent operations | Generalized binary operations | Alternative binary operations generate systems with new structural properties. | Supports redefining arithmetic operations through a parameter. |
| 3 | Grishkov and Zavarnitsine [17] | Non-associative structures | Groups with triality | Structural identities may replace classical associativity while preserving algebraic consistency. | Related to parameter-dependent structural behavior. |
| 4 | Hall [18] | Non-associative structures | Moufang loops and triality | Studies algebraic systems where symmetry and special identities replace standard associativity. | Conceptually related to generalized algebraic operations. |
| 5 | Marouf et al. [24] | Fast exponentiation methods | Modular arithmetic | Comparative study of binary methods, Montgomery ladder, sliding-window, NAF, MOF, and w-NAF. | Provides classical foundations of efficient exponentiation methods. |
| 6 | El Makkaoui et al. [20] | Fast exponentiation methods | Public-key cryptography | Reviews practical modular exponentiation methods for cryptographic systems. | Shows the practical importance of efficient exponentiation. |
| 7 | El Makkaoui et al. [19] | Cryptographic optimization | RSA cryptosystem | Fast RSA variants reduce the cost of decryption and digital signature generation. | Confirms exponentiation efficiency remains critical in real systems. |
| 8 | Doukas [23] | Specialized exponentiation | Galois fields GF($2^n$) | Fast squaring, efficient multiplication, and precomputation significantly accelerate exponentiation. | Demonstrates optimization in alternative algebraic environments. |
| 9 | Aggarwal and Isaacs [15] | Asymptotic improvement | Factored modulus arithmetic | Faster modular exponentiation is achieved by exploiting factorized modulus structure. | Indicates exponentiation remains an active theoretical research topic. |
| 10 | Somsuk et al. [25] | Parallel acceleration | Modular arithmetic | Improved modular exponentiation using periodic bit-pattern exponents and acceleration techniques. | Shows that implementation-level optimization is still actively studied. |

The efficiency of parametric exponentiation directly affects the practical performance of these schemes. Therefore, when parametric algebra is applied in cryptographic systems, not only security but also the computational efficiency of algebraic operations becomes important. This requires a separate consideration of existing parametric cryptosystems and the role of parametric exponentiation within them. The next section is devoted precisely to these issues.

### C. Existing Parametric Cryptosystems and Research Gap

Several cryptographic algorithms have been proposed based on parametric algebra, and some of them have reached the level of standardization. Symmetric, public-key, hash-function, and digital signature systems enhanced by parametric algebra are presented in Fig. 1. The figure also shows the hierarchical use of parametric operations in key generation, encryption or signing, and decryption or verification stages [5-11].

As shown in Fig. 1, parametric multiplication, parametric inverse operations, and parametric exponentiation are used as core computational components in different cryptographic mechanisms. In particular, parametric exponentiation appears in several stages of public-key systems and digital signature schemes, which highlights its practical importance.

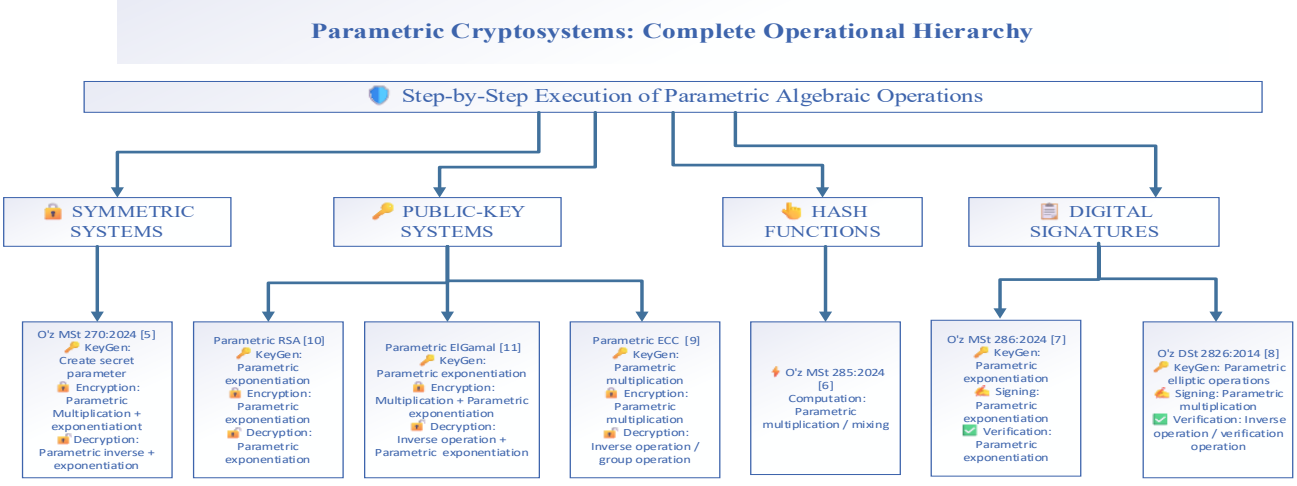


Fig. 1. Parametric cryptosystems

Existing parametric cryptosystems show that parametric exponentiation is a central operation in many schemes. However, in most cases it is computed by iterative or sequential procedures based on repeated parametric multiplications. This leads to increased computational cost, especially for large exponents and parameter values, motivating the need for more efficient methods.

As a result, expanding the practical applicability of parametric algebraic systems requires optimization of their core operations, especially exponentiation. Therefore, developing efficient parametric exponentiation methods remains an important research problem.

## III. Problem Statement

In parametric algebraic systems, one of the main operations is parametric multiplication, which is defined as follows:

$$a®b \equiv a + b + a * R * b \ (mod \ \ n),$$

where, $R$ is an additional parameter of the algebraic structure. Based on this operation, parametric exponentiation is widely used in existing cryptographic algorithms. In practice, it is usually represented in the following form:

- Initial condition: $a^{\backslash 1} = a \ mod \ n$.
- Recursive step: $a^{\backslash x+1} = a + a^{\backslash x}(1 + Ra) \ mod \ n$

A direct algorithm for computing this expression can be written as follows (see Algorithm 1):

**Algorithm 1:** Existing Iterative Method for Parametric Exponentiation

```
Input: Integers a, k, R, n
Output: a^(k) mod n
1: if k = 1 then
2: return a mod n
3.  end if
4. t ← (1 + R·a) mod n
5: for i = 2 to k do
6:    result ← (a + result·t) mod n
7:    end for
8: return result
```

This algorithm is a simple and direct computation method. However, the parametric multiplication operation is performed sequentially $k - 1$ times. Therefore, the computational cost increases as the exponent value grows.

In addition, since all operations depend on the parameter $R$, the execution time may become more significant for large parameter values.

Several methods exist for computing parametric exponentiation in systems based on parametric algebra. These include recursive iterative methods, sequential procedures based on repeated parametric multiplication, and summation or closed-form formulas. However, practical computational efficiency remains an important issue.

Thus, the main problem is to develop a more efficient method for computing parametric exponentiation while preserving the correctness of the algebraic system.

## IV. Parametric Algebra and its Properties

Since parametric algebra is a relatively new algebraic framework in cryptographic research and remains less familiar to the broader community, its main operations, fundamental properties, and illustrative examples are presented in this section.

Parametric algebra extends classical algebraic operations by introducing an additional parameter. This parameter increases computational complexity and thereby enhances cryptographic resilience. Consequently, it provides sufficient security for encryption algorithms based on parametric algebra in large finite fields.

Parametric algebra proposes an extended form of classical arithmetic operations, serving as a promising foundation for improving the mathematical structure of cryptographic algorithms and incorporating an additional complexity component [3].

In *parametric algebra*, the parameter $R$ is selected from the set of numbers in the field $Z_p = \{0,1, \dots, n-1\}$ and all operations are performed modulo $n$.

### A. Multiplication with Parameter R

**Definition 1.** For any $a, b \in Z$, the expression:

$$a®b \equiv a + b + a * R * b \ (mod \ \ n)$$

defines the *parametric multiplication operation* with parameter $R$. In the context of parametric algebra, the symbol $R$ is referred to as either a *coefficient* or a *transformation parameter*. When $R = 0$, this expression reduces to the *classical algebraic addition* operation [3].

**Example:** If $a = 3, b = 5, R = 2, n = 17$ then,

$$3 \ ® \ 5 \ mod \ n = 3 + 5 + 3 \cdot 2 \cdot 5 \ mod \ 17 = 38 \ mod \ 17 = 4$$

This operation serves as the fundamental functional unit of the proposed parametric algebraic structure.

The parametric multiplication operation defined above satisfies the following properties:

**Closure**. For arbitrary $a, b \in Z_n$, the operation is closed:

$$a \ ® \ b = c \ mod \ n, c \in Z_n$$

**Commutativity**. The operation is commutative, as $a \ ® \ b = b \ ® \ a \ mod \ n$ for all $a, b, R \in Z_n$. This result follows from the symmetry of the given expression:

$$a + b + a \ \cdot R \ \cdot b = a + b + \ b \cdot R \ \cdot a$$

**Associativity**. The operation is associative for all $a, b, R \in Z_n$: $(a \ ® \ b)® \ c \ mod \ n = a \ ® \ (b \ ® \ c) mod n$.

**Identity Element.** There exists an identity element $e = 0$, where $a®e = a + 0 + a \cdot R \cdot 0 = a \ mod \ n$ for any $a, e, R \in Z_n$

**Inverse Element**. For each $a \neq 0$ such that $\gcd(a + aR, n) = 1$ , there exists an inverse element $a^{-1}$ where $a \ ® \ a^{-1} = e \ mod \ n$.

### B. Inversion Operation in Parametric Algebra

Parametric inversion operation modulo $n$ with parameter $R$ is defined as:

$$a^{\backslash -1} \equiv -a * (1 + R * a)^{-1} \ (mod\ n)$$

Here, the notation $^{-1}$ denotes the classical modular inversion. The element $^{\backslash -1}$ represents the parametric inverse with respect to the parameter $R$ and modulus $n$ and satisfies the identity $a \circledR a^{-1} \equiv 0 (mod\ n)$ [26].

*Verification.*

To verify this, we substitute the expression $a^{\backslash -1} \equiv -a * (1 + R * a)^{-1} \ (mod\ n)$ into the parametric multiplication formula $a \circledR a^{\backslash -1} = a + a^{\backslash -1} + a \cdot R \cdot a^{\backslash -1} mod\ n$:

$a \circledR a^{\backslash -1} mod\ n = a + (-a(1 + R * a)^{-1}) + a \cdot R \cdot (-a(1 + R * a)^{-1})\ mod\ n = a - a(1 + R * a)^{-1} - a \cdot R \cdot a(1 + R * a)^{-1}\ mod\ n = a - a(1 + R * a)^{-1}(1 + R * a)\ mod\ n$

Since $-a(1 + R * a)^{-1}(1 + R * a)\ mod\ n = 1$, we obtain $a - a \equiv 0\ mod\ n$.

**Example:** Let $a = 4, R = 2, n = 17$. First, we calculate the parametric inverse using the formula $a^{\backslash -1} \equiv -a * (1 + R * a)^{-1} \ (mod\ n)$:

*Calculate* $1 + aR = 1 + 4 \cdot 2\ mod\ 17 = 9$ mod 17

*Find the modular inverse*: $(1 + R * a)^{-1}\ mod n = 9^{-1} mod\ 17 = 2\ mod\ 17$

*Compute the final result*: $a^{\backslash -1} = -a * (1 + R * a)^{-1} = -4 \cdot 2 = -8\ mod\ 17 = 9\ mod\ 17$

*Verification*: $a \circledR a^{-1} mod\ n = 4 \circledR 9\ mod\ 17 = 4 + 9 + 4 \cdot 2 \cdot 9\ mod\ 17 = 13 + 72\ mod\ 17 = 85\ mod 17 = 0$

The inversion operation defined above establishes a robust algebraic framework. The following properties define the order and cardinality of the resulting parametric multiplicative group:

**Property 1.** If the modulus $n$ is a prime number and the parameter $R$ of the parametric multiplicative group is any natural number, then the order of the group is equal to $\varphi(n)$, where $\varphi(n)$ represents the value of *Euler's totient function*.

**Property 2.** For a parametric multiplicative group with a composite modulus $n$ if the parameter $R$ is coprime to the modulus $(\gcd(R, n) = 1)$, then the order of the group is equal to $\varphi(n)$, where $\varphi(n)$is the value of Euler's totient function.

**Implications for Inversion Operations.** These properties are critical for the inversion process. The group order $\varphi(n)$$ ensures that for every valid element, a unique inverse exists within the structure. Furthermore, according to the parametric adaptation of Euler's Theorem, the inverse can also be computed through exponentiation:

$$a^{-1} = a^{\backslash \varphi(n) - 1}\ mod\ n$$

This dual path for inversion enhances the cryptographic flexibility of the proposed system.

### C. Parametric Exponentiation Operation

The concept of parametric exponentiation is based on the repeated application of the parametric multiplication operation.

**Definition 2**. In a parametric multiplicative group modulo $n$ with a parameter $R \geq 1$, the operation of exponentiation is referred to as a parametric exponentiation function.

The result of raising a base $a$ to the power $x$ is defined by the following recursive relation:

Initial condition: $a^{\backslash 1} = a\ mod\ n$.

Recursive step: $a^{\backslash x+1} = a + a^{\backslash x}(1 + Ra)\ mod\ n$

where, $x \in N$ and the element $^{\backslash 1}$ denotes the parametric exponentiation operator [3].

**Example.** Let $n = 17, R = 2, a = 3$ and $k = 4$. We aim to calculate the parametric exponentiation value $y = 3^{\backslash 4}\ mod 17$.

First, calculate the transform factor $F$:

$F = 1 + R \cdot a\ mod\ n = 1 + 2 \cdot 3\ mod 17 = 7\ mod 17$

Next, the sequence is recursively defined by $d_1 = a, d_{t+1} = a + F \cdot d_t\ mod\ n$.

The sequence $d_t$ represents the iterative computation of parametric exponentiation.

$d_1 = a = 3,$

$d_2 = a + Fd_1\ mod\ n = 3 + 7 \cdot 3 = 3 + 21\ mod 17 = 7\ mod 17$

$d_3 = 3 + 7 \cdot 7 = 3 + 49\ mod 17 = 1\ mod 17$

$d_4 = 3 + 7 \cdot 1 = 3 + 7\ mod 17 = 10\ mod 17$

The iterative calculation yields $3^{\backslash 4}\ mod 17 = 10\ mod 17$, which confirms the result of the parametric power function for the given parameters.

The parametric exponentiation function exhibits several fundamental properties as outlined below [3].

**Property 1.** For arbitrary $a, z, d \in \mathbb{Z}_n$, the following relations hold:

$a^{\backslash z+d} = a^{\backslash z} \circledR a^{\backslash d} mod\ n,\ a^{\backslash z} = a^{\backslash z} \circledR e\ mod\ n$

where, $\circledR -$ denotes the parametric multiplication operation modulo $n$, $e$ is the identity element and $a^{\backslash x}$ - denotes the parametric exponentiation operator.

**Property 2.** For arbitrary $a, z, d \in \mathbb{Z}_n$, the following relations hold: $a^{\backslash z*d} = (a^{\backslash z})^{\backslash d} = (a^{\backslash d})^{\backslash z}\ mod\ n$

where, $a^{\backslash x}$ - denotes the parametric exponentiation operator.

**Property 3**. For any $a \in \mathbb{Z}_n$, the following holds:

$a^{\varphi(n)+1} = a\ mod\ n,\ a^{\backslash 1} = a$, where $\varphi(n)$denotes Euler's totient function.

**Property 4**. If $d$ and $e$ are coprime to $\varphi(n)$ and satisfy the condition $d \cdot e \equiv 1\ mod\ (\varphi(n))$, then the following relation holds: $(a^{\backslash d})^{\backslash e} = a\ mod\ n$, where $a \in \mathbb{Z}_n$ and $a^{\backslash x}$ - denotes parametric exponentiation operator.

**Property 5**. For any $a \in \mathbb{Z}_n$, the following relations hold:

$$a^{\backslash 0} = 0, a^{\backslash -1} = -a * (1 + R * a)^{-1} \ mod\, n$$

where, $a^{\backslash -1} -$ represents the parametric inverse of $a$ modulo $n$ and $(1 + R * a)^{-1}$ denotes the classical modular inverse of $(1 + R * a)$ modulo $n$.

**Property 6.** The relationship between parametric exponentiation and classical modular exponentiation is given by

$$R * a^{\backslash x}\ (mod n) = (1 + R * a)^x - 1\ mod\ n$$

where $R$ is a parameter and $\neq 0$ , $\in \{1,2,.\varphi(n) - 1\}$, $a \in \{1,2,..,n-1,$ $a^{\backslash x}$ - denotes parametric exponentiation operator.

**Property 7.** Let $F = 1 + Ra$ . Then, for parametric exponentiation,

$$a^{\backslash x+1} \equiv a * \sum_{i=0}^{i=x} F^{\,i} (mod\ n)$$

where, $R$ is a parameter, $a^{\backslash x+1}$ - denotes the parametric exponentiation operator, $\sum_{i=0}^{i=x} F^{\,i} (mod\ n)$ denotes the geometric sum.

The algebraic properties presented above facilitate the enhancement of existing cryptographic algorithms and enable more efficient computational processes within parametric structures.

For a more comprehensive discussion and additional definitions of parametric algebraic operations, the reader is referred to [3].

## V. Efficiency Analysis of Parametric Exponentiation

In this section, the computational efficiency of parametric exponentiation is analyzed. Various approaches, including iterative and closed-form methods, are examined. In addition, methods for improving algorithmic efficiency are proposed by applying classical modular binary exponentiation techniques to existing algorithms.

Several publications and papers on parametric algebra present both iterative (sequential) and closed-form methods for exponentiation with parameter $R$ [3, 4, 12].

### A. Iterative Parametric Exponentiation

Let the parametric multiplication operation modulo $n$ be defined as follows:

$$a®b \equiv a + b + a * R * b\ (mod\ \ n),$$

Where $R \in Z_n$ is a parameter, $a, b \in Z_n$ . This operation differs from classical multiplication and represents a combination of addition and multiplication.

The exponentiation operation based on this parametric multiplication is defined as:

$$a^{\backslash k} = \underbrace{a®a®a® \ldots . a®}_{k\ times}$$

Specifically, $a^{\backslash 1} = a$, $a^{\backslash k+1} = a^k ® a$. This expression is an analogous form of the classical exponentiation operation, where the parametric multiplication is utilized instead of the standard multiplication.

Parametric exponentiation satisfies the following recursive relation:

$$a^{\backslash k+1} = a^{\backslash k} ® a$$

Using the definition of the parametric multiplication, we obtain $a^{\backslash k+1} = a^{\backslash k} + a + a^{\backslash k} \cdot R \cdot a\ mod n$

This expression can be rewritten as follows:

$$a^{\backslash k+1} = a + a^{\backslash k}(1 + Ra) mod\ n$$

Let $d_k = a^{\backslash k}$ Then, the recurrence relation can be written as

$$d_{k+1} = a + d_k(1 + Ra) mod\ n \quad (3)$$

This recurrence defines the iterative computation of parametric exponentiation up to the exponent $k$.

By leveraging the algebraic properties of parametric exponentiation, the recurrence relation, Formula (3), can be transformed into equivalent alternative representations. These forms facilitate further optimization and significantly enhance computational efficiency.

### B. Alternative Forms of Parameteric Exponentiation

*Based on Property 7* of parametric exponentiation, the following *summation-based representation* is derived:

$$a^{\backslash x+1} \equiv a * \sum_{i=0}^{i=x} F^{\,i} (mod\ n). \quad (4)$$

Based on Formula (4), the exponentiation operation can be computed without relying on repeated squaring, by evaluating a geometric series and performing a single modular multiplication with the base $a$. This approach reduces the number of modular multiplications, enables efficient precomputation, and eliminates bit-dependent branching. Consequently, it yields a more efficient exponentiation mechanism for cryptographic algorithms based on parametric algebra.

The closed-form expression Formula (4) can be *derived from the recurrence relation* Formula (3) as follows:

According to Formula (3), we have $d_1 = a$ , $d_2 = a + d_1(1 + Ra)$, $d_3 = a + d_2(1 + Ra)$,…

To simplify the expression, the following notation is introduced:

$$F = 1 + Ra$$

Then, the recurrence relation can be written as

$$d_{k+1} = a + F \cdot d_k mod\ n$$

Expanding this recurrence iteratively yields:

$$d_1 = a$$

$$d_2 = a + Fa$$

$$d_3 = a + F(a + Fa) = a + Fa + Fa^2$$

$$d_4 = a + F(a + Fa + Fa^2) = a + Fa + F^2a + F^3a$$

Continuing this pattern, the general term for $d_k$ is expressed as:

$$d_k = a(1 + F + F^2 + \cdots + F^x)$$

Consequently,

$$a^{\backslash k+1} = a\sum_{i=0}^{k} F^i \ mod\ n, F = 1 + Ra$$

This representation corresponds exactly to Formula (4), derived from *the 7th property* of parametric exponentiation.

*Based on the 6th property* of parametric exponentiation:

$$R * a^{\backslash k}\ (mod n) = (1 + R * a)^k - 1\ mod\ n$$

it is possible to derive the *closed-form expression* for parametric exponentiation:

$$a^{\backslash k} = ((1 + Ra)^k - 1) \cdot R^{-1}\ mod\ n \qquad (5)$$

In this context, it is necessary to satisfy the condition $\gcd(R, n) = 1$ to ensure the existence of the modular inverse of the parameter $R$.

*The derivation of the closed-form Formula* (5) from the recurrence relation Formula (3) is expressed as follows.

$$\text{Let } F = 1 + Ra.$$

Substituting this into Formula (3), the recurrence relation takes the following form:

$$d_{k+1} = a + F \cdot d_k mod\ n$$

Expanding this recurrence iteratively yields:

$$d_1 = a$$

$$d_2 = a + Fa$$

$$d_3 = a + F(a + Fa) = a + Fa + Fa^2$$

$$d_4 = a + F(a + Fa + Fa^2) = a + Fa + F^2a + F^3a$$

$$.d_k = a(1 + F + F^2 + \cdots + F^{k-1}) mod\ n$$

This sequence represents a geometric progression. From standard mathematics, the sum of a geometric series is given by:

$$x + x \cdot q + x \cdot q^2 + \cdots + x \cdot q^{k-1} = x \cdot \frac{q^k - 1}{q - 1}$$

In our case, $x = a, q = 1 + Ra$. Substituting these values, we derive:

$$d_k = a \cdot \frac{(1 + Ra)^k - 1}{(1 + Ra) - 1} = a\frac{(1 + Ra)^k - 1}{Ra}$$

$$= ((1 + Ra)^k - 1)\ R^{-1}\ mod\ n$$

Hence,

$$a^{\backslash k} = ((1 + Ra)^k - 1) \cdot R^{-1}\ mod\ n$$

This coincides with the closed-form expression given in Formula (5).

Although the resulting parametric exponentiation expressions provide non-iterative closed-form representations, they still require further refinement to achieve high computational efficiency. Therefore, additional optimization techniques are necessary to further accelerate their evaluation.

Efficient exponentiation techniques play a crucial role in modern cryptographic computations. Among them, binary exponentiation is one of the most widely used methods due to its logarithmic complexity.

### C. Efficient Computation Using Binary Exponentiation

For given $a, k, n \in \mathbb{N}$, one of the most efficient methods for computing $a^k mod n$ is binary exponentiation. This method is based on representing the exponent $k$ in its binary form:

$$k = \sum_{i=0}^{t} b_i 2^i,\ b_i \in \{0,1\}$$

Accordingly, the exponentiation can be expressed as

$$a^k = \prod_{i=0}^{t} a^{b_i 2^i}$$

The computation is performed by squaring and multiplying successively, with modular reduction applied at each step to prevent the intermediate values from becoming too large.

The pseudocode for this algorithm is as follows (see Algorithm 2):

**Algorithm 2:** Binary Modular Exponentiation

```
Input: Integers a, k, n
Output: a^k mod n
1: result ← 1
2: base ← a mod n
3: while k > 0 do
4:    if k mod 2 = 1 then
5:          result ← (result · base) mod n
6:    end if
7:    base ← (base · base) mod n
8:    k ← ⌊k / 2⌋
9: end while
10: return result
```

The algorithm relies on the binary representation of the exponent $k$, enabling efficient computation through successive squaring and conditional multiplication. Modular reduction at each step prevents the growth of intermediate values. With a computational complexity of $O(\log k)$, this method is widely regarded as one of the most efficient techniques in cryptographic applications.

### D. Integrating Binary Algorithms into Parametric Algebra

The binary exponentiation technique can be applied to different representations of parametric exponentiation. In particular, it can be integrated with both the iterative formulation Formula (3) and the closed-form representation Formula (5).

The *recursive form of parametric* exponentiation is given by Formula (3):

$$d_{k+1} = a + d_k(1 + Ra)(\text{mod}n).$$

This formulation enables a sequential computation of parametric exponentiation. However, it exhibits linear time complexity in the exponent $k$, namely $O(k)$.

To improve computational efficiency, the classical binary exponentiation technique can be applied to the given recurrence

process. In this approach, the exponent $k$ is represented in its binary form:

$$k = \sum_{i=0}^{t} b_i 2^i, b_i \in \{0,1\}.$$

Based on this representation, parametric exponentiation can be expressed as:

$$a^{\backslash k} = \underbrace{a®a®a® \ldots . a®}_{k\ times}$$

where, the parametric multiplication is defined as:

$$a®b = a + b + aRb(\mathrm{mod}n).$$

The computation follows the structure of binary exponentiation, involving successive squaring and conditional multiplication according to the binary representation of the exponent**.**

As a result, the following iterative algorithm is obtained. If the current bit of $k$ is equal to 1, the result is updated using parametric multiplication, while at each step the base is updated through parametric squaring.

This approach reduces the number of computational steps to $O(\log k)$.

Therefore, the application of binary exponentiation significantly reduces the number of iterations required for computing parametric exponentiation.

The pseudocode for this algorithm is as follows (see Algorithm 3):

**Algorithm 3:** Iterative Binary Parametric Exponentiation

```
Input: Integers a, k,R, n
Output: a^k mod n
1: result ← 0
2: base ← a mod n
3: while k > 0 do
4:    if k mod 2 = 1 then
5:          result ← (result + base + R · result · base) mod n
6:    end if
7:    base ← (base + base + R · base · base) mod n
8:    k ← ⌊k / 2⌋
9: end while
10: return result
```

**Example.** Let, $R = 19$, $a = 5$, and $k = 13$.

We compute the value of parametric exponentiation.

Binary representation:

$$k = 13 = (1101)_2$$

The parametric multiplication is defined as

$$a®b = a + b + aRb(\mathrm{mod}n).$$

Initial values:

$$\mathrm{result} = 0, \mathrm{base} = 5.$$

**Step 1** ($k = 13$, bit = 1)

$$\mathrm{result} = 0®5 = 5$$

$$\mathrm{base} = 5®5 = 5 + 5 + 19 \cdot 25 = 485 \equiv 23(\mathrm{mod}77)$$

**Step 2** ($k = 6$, bit = 0)

$$\mathrm{result} = 5$$

$$\mathrm{base} = 23®23 = 23 + 23 + 19 \cdot 529 = 10097 \equiv 10(\mathrm{mod}77)$$

**Step 3** ($k = 3$, bit = 1)

$$\mathrm{result} = 5®10 = 5 + 10 + 19 \cdot 50 = 965 \equiv 41(\mathrm{mod}77)$$

$$\mathrm{base} = 10®10 = 10 + 10 + 19 \cdot 100 = 1920 \equiv 72(\mathrm{mod}77)$$

**Final result:**

$$a^{\backslash 13} \equiv 68(\mathrm{mod}77).$$

Formula (5), derived from Property 6 of parametric exponentiation, transforms the parametric exponentiation into a classical modular exponentiation form. In this formulation, the main computational cost is associated with evaluating:

$$F^k = (1 + Ra)^k.$$

Since this expression corresponds to classical exponentiation, it can be efficiently computed using standard modular exponentiation techniques.

Thus, parametric exponentiation can be reduced to a classical modular exponentiation problem via Formula (5), where the value $F^k$ is efficiently computed using binary exponentiation.

The pseudocode for computing parametric exponentiation using the binary exponentiation method is as follows (see Algorithm 4):

**Algorithm 4:** Binary Parametric Exponentiation Closed-form

```
Input: Integers a, k, R, n
Output: a^k mod n
1: if k = 0 then
2: return 0
3: while k > 1 do
4:    A ← (1 + R · a)mod n
5:    invR ← R^{-1}modn  //assuming gcd(R,n)=1
6:    t ← BinaryExponantiation (A, k, n)
7:    result ← (t − 1)mod n
8:    result ← (result · invR)modn
9: return result
```

**Example.** Given the values $n = 77, R = 19$, $a = 5$, and $k = 13$ , compute $a^{(k)}\mathrm{mod}\ n$ using the closed-form representation Formula (5).

**Step 1.** Using Formula (5)

$$a^{\backslash k} = ((1 + Ra)^k - 1) \cdot R^{-1}\ mod\ n$$

**Step 2.** Compute $A = 1 + Ra\mathrm{mod}\ n$

$$A = 1 + 19 \cdot 5\ mod\ 77 = 19\ mod\ 77$$

**Step 3.** Compute the modular inverse of $R$

$$R^{-1} = 19^{-1} \equiv 73 (\mathrm{mod} 77).$$

**Step 4.** Compute $19^{13}$ $mod$ 77using binary exponentiation.

$$13 = 1101_2 = 2^3 + 2^2 + 2^0$$

$$19^2 \equiv 53, 19^4 \equiv 37, 19^8 \equiv 60 (\mathrm{mod} 77)$$

$$19^{13} \equiv 19^8 \cdot 19^4 \cdot 19 \equiv 60 \cdot 37 \cdot 19 \equiv 61 (\mathrm{mod} 77)$$

**Step 5.** Final computation

$$a^{(13)} = (61 - 1) \cdot 73 = 60 \cdot 73 = 4380 \equiv 68 (\mathrm{mod} 77).$$

The analysis of the presented example demonstrates that the considered parametric exponentiation algorithms, based on Formula (3) and Formula (5), improve computational efficiency compared to the direct iterative approach.

Simple iterative parametric exponentiation method requires 13 parametric multiplications; both algorithms leverage binary exponentiation to reduce the number of iterations to ($\log k$).

Moreover, the approach based on the closed-form representation Formula (5) further simplifies the computation by reducing it to a single modular exponentiation, resulting in improved practical efficiency.

The classical modular exponentiation and the parametric methods based on Formula (3) and Formula (5) differ significantly in their algebraic foundations and computational processes. A generalized comparison of the characteristics of these exponentiation algorithms is presented in Table II.

TABLE II. PARAMETRIC CRYPTOGRAPHIC ALGORITHMS

| Property | Classical | Formula (3) | Formula (5) |
|---|---|---|---|
| Algebraic basis | Classical modular arithmetic | Parametric algebra | Parametric algebra |
| Main operation | Modular multiplication | Parametric multiplication | Modular multiplication |
| Dependence on parameter | None | Strong | Weak |
| Computational process | Squaring and multiplication | Iterative parametric multiplication | Binary exponentiation |
| Number of parametric operations | 0 | ≈ k times | 0 |
| Algorithmic complexity | O(log k) | O(k) | O(log k) |

## VI. EXPERIMENTAL RESULTS AND DISCUSSION

To improve the efficiency of parametric exponentiation, the fast binary exponentiation technique was applied to the parametric exponentiation methods considered in this study. The performance of the considered algorithms was evaluated using three metrics: *execution time*, *memory consumption*, and *CPU ticks*, in order to provide a broader view of their practical efficiency.

The performance of the considered exponentiation algorithms was evaluated with respect to the modulus $n$and the parameter $R$, in order to analyze their impact on computational efficiency.

To this end, software implementations of three algorithms were developed: the *classical* modular exponentiation algorithm, the *iterative* parametric exponentiation method based on Formula (3), and the optimized *closed-form* parametric exponentiation algorithm based on Formula (5).

For clarity, these algorithms are denoted as *Classic*, *Param5*, and *Param7*, respectively.

The software modules were implemented in the C# programming language and executed on the .NET platform.

To ensure the reliability of the performance metrics, the experiments were conducted on a computing system with the following technical specifications: Windows 11 Pro operating system, Intel Core i7-13700U processor with a clock frequency of 2.40 GHz, and 16 GB of RAM.

To ensure the reliability of the results, each algorithm was executed multiple times for each set of parameter values. The exponentiation operation was repeated 100 times within the program and the execution time was measured using a high-precision timer.

### A. Impact of the Modulus $n$

The performance of the algorithms was evaluated for different lengths of the modulus $n$. The parameters used in the computations were fixed as $R$=2048 bit, $a$=512 bit and k=256 bit.

The modulus length was gradually increased starting from 512 bits, and for each tested value, the algorithms were evaluated in terms of execution time, memory consumption, and CPU ticks.

The obtained results, illustrating the dependence of execution time on the modulus bit-length, are presented graphically in Fig. 2.

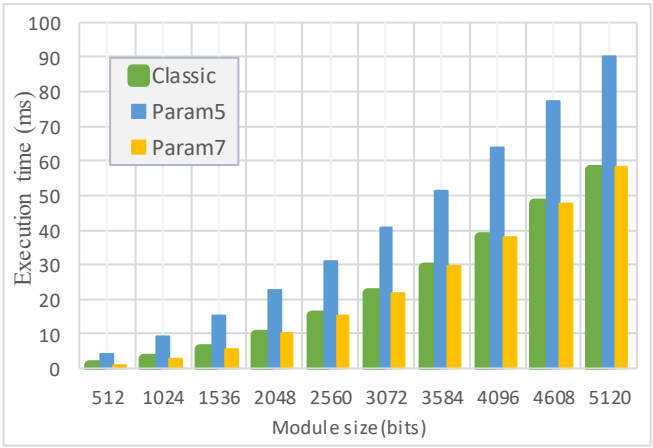


Fig. 2. Execution time vs. modulus length.

The graphical results show that the execution time of all algorithms increases as the modulus bit-length grows. The iterative parametric exponentiation method based on Formula (3) operates more slowly than the classical algorithm, since the parametric multiplication is performed multiple times.

In contrast, the exponentiation method based on Formula (5) achieves nearly the same performance as the classical algorithm.

This demonstrates that the use of a closed-form expression is effective for optimizing parametric exponentiation operations.

This behavior can be explained by the fact that the closed-form formulation eliminates repeated parametric multiplications, thereby reducing computational overhead and improving scalability.

The exact numerical values corresponding to the results shown in Fig. 2 are presented in Table III.

TABLE III. EXECUTION TIME OF ALGORITHMS FOR DIFFERENT MODULUS SIZES.

| n (bit) | Classic (ms) | Param5 (ms) | Param7 (ms) |
|---|---|---|---|
| 512 | 1.01 | 4.21 | 0.74 |
| 1024 | 2.64 | 9.32 | 2.66 |
| 2048 | 9.85 | 22.49 | 9.96 |
| 4096 | 37.76 | 63.66 | 38.16 |
| 5120 | 57.57 | 90.16 | 58.25 |

The table shows that the algorithm based on Formula (3) is slower than the classical method, whereas the one based on Formula (5) has nearly the same speed.

In addition to runtime measurements, memory usage and CPU ticks were also evaluated for different modulus sizes. The corresponding results are presented in Table IV.

TABLE IV. MEMORY USAGE AND CPU TICKS FOR DIFFERENT MODULUS SIZES

| n (bit) | Classic | | Param5 | | Param7 | |
|---|---|---|---|---|---|---|
| | Memory(bytes) | CPU Ticks | Memory(bytes) | CPU Ticks | Memory(bytes) | CPU Ticks |
| 512 | 112 | 21240 | 112 | 35332 | 112 | 6300 |
| 1024 | 112 | 21152 | 112 | 70439 | 112 | 21387 |
| 2048 | 112 | 74823 | 0 | 162860 | 112 | 75852 |
| 4096 | 0 | 283699 | 0 | 464917 | 112 | 287081 |
| 5120 | 112 | 446870 | 0 | 665616 | 112 | 442620 |

As shown in Table IV, the iterative method based on Formula (3) consistently requires more CPU ticks than the classical and closed-form methods across all tested modulus sizes. In contrast, the method based on Formula (5) remains very close to the classical algorithm, while memory usage stays nearly constant for all approaches.

The obtained results further confirm that the computational overhead associated with iterative parametric exponentiation significantly increases execution time and CPU ticks, whereas the closed-form approach mitigates this effect while preserving low memory usage.

### *B. Impact of the Parameter R*

The performance of the algorithms was evaluated for different bit-lengths of the parameter $R$. This allows assessment of the effect of parameter size on execution time, memory usage, and CPU ticks. The values used in the computations were fixed as $n = 4096$bits, $a = 512$bits, and $k = 256$bits.

The bit-length of parameter $R$ was gradually increased starting from 128 bits, and for each value, the execution time of the algorithms was measured. The obtained results are presented graphically in Fig. 3.

As the graph shows, the execution time of the classical exponentiation algorithm remains unchanged since it is independent of the parameter $R$. By contrast, the computation time of the exponentiation algorithm based on Formula (3) increases as the value of $R$ grows.

Meanwhile, the parametric exponentiation algorithm based on Formula (5) performs similarly to the classical algorithm.

This indicates that the proposed method is largely independent of the parameter R, as the computation is reduced to a single modular exponentiation that does not involve repeated parameter-dependent operations.

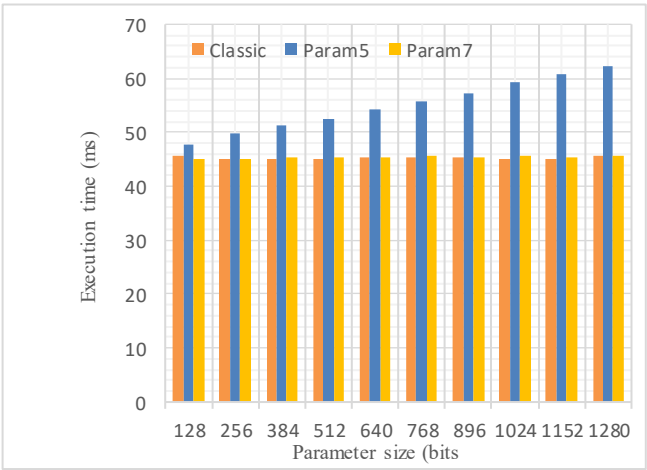


Fig. 3. Execution time vs. parameter length.

The exact numerical values corresponding to the results shown in Fig. 3 are presented in Table V.

TABLE V. EXECUTION TIME OF ALGORITHMS FOR DIFFERENT PARAMETER VALUES.

| R (bit) | Classic (ms) | Param5 (ms) | Param7 (ms) |
|---|---|---|---|
| 128 | 38.536 | 40.872 | 38.678 |
| 256 | 38.595 | 42.447 | 38.563 |
| 512 | 38.373 | 45.497 | 38.609 |
| 1024 | 38.386 | 52.356 | 38.721 |
| 1280 | 38.424 | 55.575 | 38.711 |

As can be seen from the table, the execution time of the algorithm based on Formula (3) increases with the growth of the parameter $R$. In contrast, the algorithm based on Formula (5) is almost unaffected by changes in the parameter value.

TABLE VI. MEMORY USAGE AND CPU TICKS FOR DIFFERENT PARAMETER VALUES

| R (bit) | Classic | | Param5 | | Param7 | |
|---|---|---|---|---|---|---|
| | Memory(bytes) | CPU Ticks | Memory(bytes) | CPU Ticks | Memory(bytes) | CPU Ticks |
| 128 | 112 | 295771 | 0 | 314064 | 112 | 295716 |
| 256 | 0 | 294361 | 0 | 326755 | 112 | 296239 |
| 512 | 0 | 295706 | 0 | 377922 | 112 | 298236 |
| 1024 | 0 | 309788 | 0 | 429285 | 112 | 308696 |
| 1280 | 0 | 309116 | 0 | 432376 | 112 | 304334 |

Some memory measurements are reported as zero because the benchmark records only additional managed heap allocations. In several runs, no extra allocation was detected during execution.

As shown in Table VI, CPU tick measurements clearly indicate that the iterative method based on Formula (3) becomes increasingly expensive as the parameter size grows. In contrast, the method based on Formula (5) remains close to the classical algorithm across all tested parameter sizes.

Overall, the results for different parameter sizes show that increasing the bit-length of $R$ has a significant impact on the iterative method based on Formula (3), leading to higher computational overhead. In contrast, the closed-form method based on Formula (5) remains close to the classical algorithm in both runtime and CPU ticks, demonstrating better scalability with respect to the parameter size.

## VII. CONCLUSION AND FUTURE WORKS

In this study, parametric algebraic operations were systematically studied, and an algorithm for parametric exponentiation based on modular binary exponentiation was proposed. By exploiting the properties of parametric algebra, a fast exponentiation expression for parametric exponentiation was derived as:

$$a^{\backslash k} = ((1 + Ra)^k - 1) \cdot R^{-1} \, mod \, n$$

As a result, the proposed method significantly reduces the computational impact of the parameter $R$ on parametric exponentiation while improving overall practical efficiency.

According to the experimental results, for modulus bit-lengths ranging from 512 to 5120 bits, the execution time of the algorithm based on Formula (5) is nearly identical to that of the classical exponentiation algorithm, with the difference not exceeding 1–2% on average.

At the same time, it was observed that the algorithm based on Formula (5) operates approximately 1.7–1.8 times faster than the optimized method based on Formula (3).

Furthermore, during the experiments, it was determined that increasing the bit-length of the parameter $R$ (from 128 bits to 1280 bits) does not significantly affect the performance of the proposed algorithm based on Formula (5).

These results confirm that parametric exponentiation can be reduced to a single modular exponentiation while preserving nearly the same performance as the classical method.

The proposed approach provides a practical and efficient alternative to classical exponentiation methods, making it suitable for high-performance cryptographic applications.

Furthermore, the proposed approach achieves a favorable balance between computational efficiency, low memory consumption, and reduced processor workload, making it suitable for practical cryptographic applications.

The computational Formula (5) proposed in this study enables efficient exponentiation in encryption algorithms and standards such as RSA and ElGamal when enhanced with parametric algebraic operations.

Limitation. This study has several limitations. Although the proposed method significantly improves the efficiency of parametric exponentiation, the practical deployment of parametric cryptosystems still requires broader optimization of all parameter-dependent operations.

Existing hardware and software acceleration mechanisms are mainly designed for classical arithmetic, while specialized implementations for parametric algebra remain limited.

In addition, the correct selection of the parameter $R$ is important because the algebraic properties of the system may depend on its value. For example, the existence of inverse elements, the applicability of the closed-form reduction in Formula (5), and the requirement to satisfy conditions such as $GCD(R, n) = 1$ may introduce additional computational overhead through GCD checks and parameter validation procedures. If these conditions are not satisfied, some algebraic operations may become invalid or require alternative computational methods.

Future work includes the implementation of the proposed method in real-world cryptographic systems and its optimization for parallel and distributed computing environments. In addition, a formal analysis of the security implications of the proposed computational reduction remains an important research direction.

Overall, the results confirm both the theoretical soundness and practical efficiency of the proposed parametric exponentiation method.

## REFERENCES

[1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. Boca Raton: CRC Press, 2018.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] X. P. Xasanov, *Takomillashgan diamatristalar algebralari va parametrli algebra asosida kriptotizimlar yaratish usullari va algoritmlari*. Toshkent: Fan, 2008. [in Uzbek]

[4] O. Ahmedova, U. Mardiyev, and O. Tursunov, "Generation and distribution of secret encryption keys with a parameter," in *Proc. International Conference on Information Science and Communications Technologies (ICISCT)*, 2020, pp. 1–4.

[5] UNICON.UZ, "Information technology. Cryptographic protection of information. Data encryption algorithm," UzNST 270:2024, 2024.

[6] UNICON.UZ, "Information technology. Cryptographic protection of information. Hash function," UzNST 285:2024, 2024.

[7] UNICON.UZ, "Information technology. Cryptographic protection of information. Processes for generating and verifying electronic digital signatures," UzNST 286:2024, 2024.

[8] UNICON.UZ, "Information technology. Cryptographic protection of information. Processes for generating and verifying electronic digital signatures based on elliptic curves," UzNST 2826:2014.

[9] O. Ahmedova, U. Mardiyev, and O. Tursunov, "Algebraic structure of parametric elliptic curves," in *Proc. International Conference on Information Science and Communications Technologies (ICISCT)*, 2021, pp. 1–3.

[10] Z. Xudoykulov and U. Xudoynazarov, "Improving the RSA public-key encryption algorithm based on parametric algebra," The Latest News and Research in Education, vol. 2, no. 12, pp. 114–122, 2025.

[11] U. Xudoynazarov, "Parametrli algebraga asoslangan el-gamal shifrlash algoritmlarini gomomorfik xususiyatini tadqiq etish," *Al-Farg'oniy avlodlari*, vol. 1, no. 4, pp. 153–157, 2023. [in Uzbek]

[12] U. Mardiyev, "Using the R parameter in a Massey–Omura protocol on elliptic curves," in *Proc. International Conference on Information Science and Communications Technologies (ICISCT)*, 2021, pp. 1–3.

[13] M. Fartitchou, H. El Marraki, L. Lafkir, A. Azzouz, K. El Makkaoui, and Z. El Allali, "Public-key cryptography behind blockchain security," in Proc. 2022 5th International Conference on Networking, Information Systems and Security (NISS), 2022, pp. 1–5.

[14] M. Gerstenhaber, "On the deformation of rings and algebras," *Annals of mathematics*, vol. 79, no. 1, pp. 59–103, 1964.

[15] A. Aggarwal and M. Isaacs, "An elementary method for fast modular exponentiation with factored modulus," arXiv preprint arXiv:2401.10497, 2024.

[16] K. Sukhorukova, "Spaces of non-additive measures generated by triangular norms," Mat. Stud., vol. 59, no. 2, pp. 215–224, Jun. 2023.

[17] A. N. Grishkov and A. V. Zavarnitsine, "Groups with triality," *Journal of algebra and its applications*, vol. 5, no. 4, pp. 441–463, 2006.

[18] J. Hall, *Moufang loops and groups with triality are essentially the same thing*. Providence: American Mathematical Society, 2019.

[19] K. El Makkaoui, Y. Lamriji, A. Beni-Hssane, Y. Maleh, and I. Ouahbi, "An overview of fast variants of the RSA cryptosystem for modern cryptography applications," EDPACS, vol. 67, no. 6, pp. 25–34, 2023.

[20] K. El Makkaoui, Y. Lamriji, I. Ouahbi, O. Nabil, A. Bouzahra, and A. Beni-Hssane, "Fast modular exponentiation methods for public-key cryptography," in Proc. 2022 5th International Conference on Advanced Communication Technologies and Networking, Dec. 2022.

[21] O. Abramkina, M. Yakubova, T. Serikov, Y. Begimbayeva, and B. Yakubov, "Implementation of lattice theory into the TLS to ensure secure traffic transmission in IP networks based on IP PBX Asterisk," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 15, no. 10, 2024.

[22] D. Swetha and S. K. Mohiddin, "Advancing quantum cryptography algorithms for secure data storage and processing in cloud computing," *International journal of advanced computer science and applications*, vol. 15, no. 9, 2024.

[23] [23] N. Doukas, "Fast exponentiation in Galois fields GF($2^n$)," International Journal of Computing, vol. 24, no. 1, pp. 81–91, 2025. https://doi.org/10.47839/ijc.24.1.3879

[24] I. Marouf, M. M. Asad, and Q. A. Al-Haija, "Comparative study of efficient modular exponentiation algorithms," Compusoft, vol. 6, no. 8, p. 2381, 2017.

[25] K. Somsuk, N. Panit, and R. Phromloungsri, "An improved algorithm for accelerating modular exponentiation with periodic bit pattern exponents," KSII Transactions on Internet and Information Systems, vol. 19, no. 4, pp. 1267–1285, 2025.

[26] K. Zarifjon and K. Umidjon, "Symmetric encryption algorithm based on a one-way function extended by parametric algebra," Stanford Database Library of American Journal of Applied Science and Technology, vol. 5, no. 12, pp. 31–36, 2025.