Available online at http://jprm.sms.edu.pk/
Journal of Prime Research in Mathematics, 21(2) (2025), 37–50

Journal of Prime Research in Mathematics

An Enhanced ElGamal Cryptosystem Based on Prime Power Moduli and Modular Key Exchange

Hayder R. Hashim*

Faculty of Computer Science and Mathematics, University of Kufa, P. O. Box 400, 54001 Al Najaf, Iraq

Abstract

In this paper, we give a security enhancement of ElGamal cryptosystem based on the use of the group of units $U(n)$, where $n = p^t$ or $2p^t$, where p is an odd prime number and t is a positive integer dynamically determined by users using a modification of the Diffie-Hellman key exchange protocol. The paper presents in details the procedures of the key generation, encryption, decryption and digital signature for the proposed cryptosystem. As applications, we apply procedures of this cryptosystem over plaintexts and implemented its algorithms in the SageMath Software. Finally, we provide results concerning its complexity evaluation, its expansion rate, and its performance timing. The results show that this system is suitable for scalable real-time encryption applications as it is indicating the successful optimization performance and robust integration of security features.

Keywords: Cryptography, public key cryptosystem, ElGamal cryptosystem, signature scheme, prime numbers, hash function, group of units, Diffie–Hellman key exchange.
2010 MSC: 11T71, 11Z05, 94A60, 11A41, 11A07.

1. Introduction

The ElGamal cryptosystem is a public key cryptosystem introduced by Taher ElGamal in 1985 [4]. The structure of this system depends on the cyclic group $\mathbb{Z}_p^*$, where p is a large prime number, and its security is based on a well known unsolvable problem in number theory called the discrete logarithm problem (DLP), that is presented by finding an integer x satisfying the congruence

$$g^x \equiv y \pmod{p},$$

where g and y are elements in the group $\mathbb{Z}_p^*$, that is generated by g. The hardness of solving this problem is based on the order of the group $\mathbb{Z}_p^*$, that is given by the Euler's totient function of p, i.e. $\varphi(p) = (p-1)$ (Euler's totient function of a positive integer n is the number of positive integers less than n and relatively

*Corresponding author
Email address: hayderr.almuswi@uokufa.edu.iq (Hayder R. Hashim)

Received : 07 July 2025*; Accepted:* 28 September 2025*; Published Online:* 01 October 2025.

prime to n). Therefore, it is very important to choose a large prime number p in order to increase the security of this cryptosystem.

An important scheme associated to ElGamal cryptosystem is the digital signature scheme, which is a cryptographic method used to authenticate the messages between senders and receivers. In fact, this scheme ensures both integrity and non-repudiation.

After 1985, Miller [19] and Koblitz [15] independently proposed the use of elliptic curves in cryptography. They use the group of points on an elliptic curve over a finite field, denoted by $E(\mathbb{F}_q)$, to form DLPs, which are believed to be significantly harder than in classical groups of similar size. Thus, the combination of ElGamal's encryption scheme with elliptic curve groups leads to the Elliptic Curve ElGamal Cryptosystem (ECEG) in which the principles of the classical ElGamal cryptosystem remain the same, but the underlying operations shift from modular exponentiation to scalar multiplication on elliptic curve points. For more details on the elliptic curves (as Diophantine equations), elliptic curve cryptography and ECEG, see e.g. [7], [8] and [16]. Many authors provided several enhancements or modifications of ECEG, see e.g. [1], [6], [13] and [14].

However the ElGamal cryptosystem has been very secure against may third parties' attacks, many authors have suggested several improvements or modifications to the structures of the encryption and decryption procedures in order to make this cryptosystem is very solid against any present and future attacks. For instance, starting from 2016 Hecht [10] generalized the ElGamal cryptosystem by using a non-commutative general linear group $\mathbb{GF}(251^8)$ in the structure of procedures. The modification gives a hard problem for searching to a subgroup membership into a non-commutative structure. In 2022, Ranasinghe and Athukorala [22] gave an improvement of ElGamal cryptosystem by using properties of the prime factorization of plaintexts and using modular exponentiation twice. They showed that this system is secure against many known attacks. Moreover, in 2022 Koppaka and Lakshmi [17] developed the ElGamal cryptosytem by combining its algorithms with hyperchaotic sequences to secure data and to reduce the times of key generation, encryption and decryption in cloud environments. Last but not least, in 2024 Amirkhanova et al. [2] introduced a novel quantum-resistant cryptosystem based on combing the ElGamal cryptosystem algorithms with a Lattice-based cryptography problem represented by the Short Integer Solution. Indeed, there are many other developments regarding the ElGamal cryptosystem, one can see e.g. [5], [11], [12], [18] and the references given there.

As mentioned earlier that increasing the order of the group increases the security of ElGamal cryptosystem, and a well know group that could have a larger order of $\mathbb{Z}_p^*$ is the abelian group of units $U(n)$, that is defined by

$$\mathbb{U}(n) = \{u \in \mathbb{Z}_n \mid \gcd(u, n) = 1\}.$$

It is proved that $U(n)$ is a cyclic if and only if $n = p^t$ or $n = 2p^t$ for p is an odd prime number and t is a positive integer. It is clear that the order of this group is given by Euler's totient function:

$$\phi(p^t) = \phi(2p^t) = p^{t-1}(p-1).$$

Moreover, there are more interesting results related to the group of units, and for latter use we list the following results (for more details about the above results and the following results and their proofs, see [9], [21] and [23]):

Lemma 1.1. *Suppose that p is an odd prime number and g is a primitive root of p (i.e. g is a generator of the group of units $U(p)$), then at least one of the integers g or $g + p$ is a primitive root of p^2.*

Lemma 1.2. *If p is an odd prime number and g is a primitive root of p^2 (i.e. g is a generator of the group of units $U(p^2)$), then g is a primitive root of p^t for $t > 2$ (i.e. g is a generator of the group of units $U(p^t)$).*

Lemma 1.3. *Let p be an odd prime number and g is a primitive root of p^t for $t \geq 2$. If g is odd, then it is a primitive root of $2p^t$. Otherwise, $p^t + g$ is a primitive root of $2p^t$.*

In this paper, we propose an enhancement of the ElGamal cryptosystem concerning the mathematical structure of its key generation algorithm, encryption and decryption algorithm and digital signature algorithm. This development is based on replacing the group $\mathbb{Z}_p^*$ by the group $U(n)$ for $n = p^t$ or $n = 2p^t$, where p is a large prime number and $t > 2$ is derived from a shared secret between communicating parties using a modification of the Diffie-Hellman key exchange protocol. This modification to the classical ElGamal cryptosystem is designed to enhance security by inserting additional complexity through the use of a group of larger order and the use of a dynamically generated modulus.

Furthermore, regarding the enhancement of the digital signature scheme of ElGamal cryptosystem, this proposed cryptosystem incorporates of a hash function (e.g. SHA-256) into the digital signature scheme. Note that, the hash functions are important in cryptography, and they are kind of signatures for a text or a data file. In fact, a hash function is one way function, that maps or compresses an arbitrary size of a text or data file into a fixed size values(called by digest or hash). Also, in the hash function, different inputs cannot have the same output. One of the well used cryptographic hash function is called by the SHA-256, that generates a unique 256-bit signature (a hash value) for a text (that has 64 hexadecimal characters).

Finally, this modification and enhancement of the classical ElGamal cryptosystem is implemented and verified using the SageMath Sofware by taking the advantage of its symbolic computation capabilities. The implementation is performed for encrypting/decrypting and signing texts and gray-scale images.

In the next section, we introduce the algorithms of the key generation, encryption, decryption and digital signature algorithm of our proposed cryptosystem. For more details about these algorithms in the classical ElGamal cryptosystem, see e.g. [4].

Remark 1.4. Since the enhanced ElGamal cryptosystem is based on the group of units $U(n)$ for $n = p^t$ or $2p^t$ with $t > 2$, in the rest of the paper we only focus on the case where $n = p^t$ and the other case can be treated similarly.

Regarding to the group of units $U(p^t)$ with $t > 2$, we see from Lemmas 1.1, 1.2 and 1.3 that if g is a generator to the groups $U(p)$ and $U(p^2)$, then it is a generator to the groups $U(p^t)$. Therefore, for the simplicity of presenting the encryption, decryption and signature scheme algorithms of our proposed cryptosystem, in the next sections we assume that a generator g to $U(p)$ is also a generator to $U(p^2)$, so that it is a generator to $U(p^t)$ where $t > 2$ is derived by a modification of the classical Diffie-Hellman key exchange protocol, that we call the Modified Diffie-Hellman key exchange protocol (MDHKEP). The MDHKEP has the exact structure of the classical Diffie-Hellman key exchange protocol with replacing the modulo p by modulo p^4), for more detail about the classical Diffie-Hellman key exchange protocol, we refer to the original article by Diffie and Hellman [3].

2. Main Phases of the Enhanced ElGamal Cryptosystem's Algorithms

Without loss of generality and in order to present the algorithms smoothly, we assume two parties presented by Alice and Bob, who want to communicate using this cryptosystem. Indeed, we assume that Bob who does the encryption with his signature on the message, and Alice does the decryption with the verification of the signature. Suppose that Bob wants to encrypt a message M (that could be a text or an image) for Alice, then they preform the following steps:

2.1. Key Generation

Since this cryptosystem is based on using the group of units $U(p^t)$, where p is a large prime number and $t > 2$ is derived from a shared secret between the communicating parties (e.g. Alice and Bob) using the Modified Diffie-Hellman key exchange protocol (MDHKEP), in this step the key generation is performed in three phases after the agreement of the public parameters. The first phase concerns generating the public and private keys, that are used in the MDHKEP. In the second phase, these parties compute the shared secret, that leads to the derivation of the value of t. Finally, they use the group of units $U(p^t)$ to generate their main public and private keys, which are used in the encryption, decryption and signature scheme procedures. These phases are described below after selecting the public parameters:

(a) **Public parameters:** Suppose that the communicating parties; Alice and Bob agree on:

- A large prime number p.
- A generator g of the groups $U(p)$ and $U(p^2)$ (g is also a generator of $U(p^t)$ for $t > 2$, see Remark 1.4).
- Fixed parameters $R, t_0 \in \mathbb{N}$ with $t_0 \geq 4$ agreed on. These are used in determining t for encryption and decryption procedures.

(b) **MDHKEP's private and public keys:**

- Alice selects a private key a_t such that $2 \leq a_t \leq p^4 - 2$.
- Bob selects a private key b_t such that $2 \leq b_t \leq p^4 - 2$.
- Alice computes $A \equiv g^{a_t} \pmod{p^4}$ and publishes the value of A.
- Bob computes $B \equiv g^{b_t} \pmod{p^4}$ and publishes the value of B.

(c) **Shared secret and the exponent t:**

- If Bob wants to send a message to Alice, he firstly has to determine the shared secret K by using Alices' public key A as follows:
$$K \equiv B^{a_t} (\text{mod } p^4).$$
- He uses the fixed parameters $R, t_0 \in \mathbb{N}$ to compute the value of t as follows:
$$t \equiv t_0 + (K \bmod R).$$
Hence, $n = p^t$ and $\varphi(n) = (p-1)p^{t-1}$, where $t \geq 4$ since $t_0 \geq 4$.

Similar steps can done by Alice using Bob's public key B to compute the exponent t.

(d) **Main private and public keys:**

- Alice selects a private keys a with $2 \leq a \leq n - 2$.
- Alice computes $r_1 \equiv g^a \pmod{n}$ and publishes (g, r_1).
- Bob selects a private keys b with $2 \leq b \leq n - 2$.
- Bob computes $r_2 \equiv g^b \pmod{n}$ and publishes (g, r_2).

2.2. Encryption (by the sender Bob)

Bob uses Alice's public key (g, r_1) and preforms the following steps:

(a) **Message preparation:** Convert the plaintext message M into a sequence of ASCII encoded integers $m_1, m_2, \ldots, m_k$.

(b) **Encryption process:** For each m_i with $1 \leq i \leq k$, choose a random ephemeral key $f_i \in [1, p^t - 2]$ to compute:
$$C_{1,i} \equiv g^{f_i} (\text{mod } n), \quad C_{2,i} \equiv m_i \cdot r_1^{f_i} (\text{mod } n)$$
The ciphertext corresponding to the plaintext message M is the ordered collection of pairs $(C_{1,i}, C_{2,i})$, for $i = 1, 2, \ldots k$.

2.3. Signature Scheme (by Bob)

Bob uses his private key b with the generator g and does the following:

(a) **Hashing M:** Given a message M (a string), the sender (Bob) performs the following steps:

- Compute its SHA-256 hash (given in hexadecimal).
- Convert the hex string to an integer.
- Reduce the resulting integer modulo $\varphi(n)$, denote the result by H.

(b) **Signature of M:**

- Choose a random integer k_s such that $\gcd(k_s, \varphi(n)) = 1$ and $1 \leq k_s \leq \varphi(n) - 2$ to compute

$$S_1 \equiv g^{k_s} (\text{mod } n), \quad S_2 \equiv {k_s}^{-1} \cdot (H - b \cdot S_1) (\text{mod } \varphi(n)). \tag{2.1}$$

- The signature pair (S_1, S_2) is sent along with the ciphertext to receiver for verification.

2.4. Decryption (by the receiver Alice)

Alice uses her main private a and preforms the following:

(a) **Decryption process:** For each ciphertext pair $(C_{1,i}, C_{2,i})$ with $1 \leq i \leq k$, compute

$$m_i \equiv C_{2,i} \cdot (C_{1,i}^a)^{-1} (\text{mod } n).$$

This can be verified as follows:

$$C_{2,i} \cdot (C_{1,i}^a)^{-1} (\text{mod } n) \equiv m_i \cdot g^{ak_i} \cdot (g^{ak_i})^{-1} \equiv m_i (\text{mod } n).$$

(b) **Message recovery:** Recover the message M from the decrypted ASCII codes m_i.

2.5. Signature Verification (by Alice)

Given a message M (a string), Alice performs the following steps:

(a) **Hashing M:**

- Compute its SHA-256 hash (given in hexadecimal).
- Convert the hex string to an integer.
- Reduce the resulting integer modulo $\varphi(n)$, denote the result by H.

(b) **Verifying the signature:** Using Bob's public key r_2, the signature pair (S_1, S_2), and the computed hash H, Alice verifies the validity of the signature by checking if the the following congruence holds, then signature is valid and accepted; otherwise it is rejected:

$$g^H \equiv r_2^{S_1} \cdot S_1^{S_2} (\text{mod } n). \tag{2.2}$$

If the congruence holds, the signature is accepted as valid; otherwise, it is rejected. Once can easily verify the validity of the above congruence as follows.

Starting from the right hand side of congruence (2.2) with the substitution of the signature construction given in (2.1), we get that

$$\begin{aligned}
r_2^{S_1} \cdot S_1^{S_2} =& r_2^{g^{k_s}} \cdot g^{k_s {k_s}^{-1} \cdot (H - b \cdot g^{k_s})} \\
=& r_2^{g^{k_s}} \cdot g^{k_s {k_s}^{-1} \cdot (H - b \cdot g^{k_s})} \\
\equiv& r_2^{g^{k_s}} \cdot g^{(H - b \cdot g^{k_s})} (\text{mod } n) \quad \text{since } g^{k_s {k_s}^{-1}} \equiv g \cdot g^{\varphi(n) w} \equiv g (\text{mod } n) \\
\equiv& g^{(b \cdot g^{k_s})} \cdot g^H g^{(-b \cdot g^{k_s})} (\text{mod } n) \quad \text{since } r_2 \equiv g^b (\text{mod } n) \\
\equiv& g^H (\text{mod } n),
\end{aligned}$$

which verifies the left hand side of congruence (2.2).

2.6. Illustrative Example

Let's illustrate a complete example of the enhanced ElGamal cryptosystem. Suppose that Bob wants to send the message `ElGamal Cryptosystem` to Alice using the enhanced ElGamal cryptosystem, they have to perform each of the above phases as follows (calculations are performed using SageMath Software [24]):

Phase I. Key generation:

(a) **Public parameters:** Suppose that Alice and Bob agree on a prime modulus $p = 11$, a generator $g = 2$, the parameters $R = 5$ and $t_0 = 4$.

(b) **MDHKEP's private and public keys:**

- Alice selects a private key $a_t = 3$.
- Bob selects a private key $b_t = 6$.
- Alice computes $A \equiv g^{a_t} (\text{mod } p^4) \equiv 2^3 (\text{mod } 14641) \equiv 8$ and publishes 8 as her MDHKEP's public key.
- Similarly, Bob's MDHKEP public key is $B = 64$.

(c) **Shared secret and the exponent t:**

- The shared secret K is obtained by both of them as follows:

$$K = B^{a_t} = A^{b_t} \equiv 262144 (\text{mod } 14641) \equiv 13247.$$

- Therefore, the exponent t is given by

$$t \equiv t_0 + (K \bmod R) \equiv 4 + (13247 \bmod 5) \equiv 6.$$

Hence, $n = 11^6 = 1771561, \quad \varphi(1771561) = 1610510.$

(d) **Main private and public keys:**

- Alice selects a private keys $a = 7$.
- Bob selects a private keys $b = 9$.
- Alice computes

$$r_1 \equiv g^a (\text{mod} n) \equiv 2^7 (\text{mod } 1771561) \equiv 128$$

and publishes $(g, r_1) = (2, 128)$ as her cryptosystem's public key.
- Similarly, Bob's public key is $(g, r_2) = (2, 512)$.

Phase II. Encryption procedure:

(a) **Message preparation:** Bob converts the plaintext message: ElGamal Cryptosystem into the following ASCII representation m_i with $1 \leq i \leq 20$:

$$\{69, 108, 71, 97, 109, 97, 108, 32, 67, 114, 121, 112, 116, 111, 115, 121, 115, 116, 101, 109\}.$$

(b) **Encryption process:** For each m_i, Bob chooses a random ephemeral key f_i, e.g.

$$\{3, 4, 2, 5, 1, 3, 4, 2, 5, 1, 3, 4, 5, 1, 3, 4, 5, 3, 2, 5\}.$$

Note that it is better to select distinct values f_i for every m_i. Then, he computes the corresponding ciphertext in the order pairs $(C_{1,i}, C_{2,i})$ using Alice's public key $(g, r_1) = (2, 128)$ and the congruences:

$$C_{1,i} \equiv g^{f_i} (\text{mod } 1771561), \quad C_{2,i} \equiv m_i \cdot r_1^{f_i} (\text{mod } 1771561).$$

In the following table, we summarize the results of computations:

Table 1: Encryption Results.

Char	m_i	f_i	$C_{1,i}$	$r_1^{f_i} \bmod n$	$C_{2,i}$
E	69	3	8	325591	1207047
l	108	4	16	929745	1205044
G	71	2	4	16384	1163264
a	97	5	32	312773	222444
m	109	1	2	128	13952
a	97	3	8	325591	1465790
l	108	4	16	929745	1205044
(space)	32	2	4	16384	524288
C	67	5	32	312773	1468620
r	114	1	2	128	14592
y	121	3	8	325591	422169
p	112	4	16	929745	1380902
t	116	5	32	312773	850448
o	111	1	2	128	14208
s	115	3	8	325591	240184
y	121	4	16	929745	890802
s	115	5	32	312773	537675
t	116	3	8	325591	565775
e	101	2	4	16384	1654784
m	109	5	32	312773	432598

Phase III. Signature scheme:

(a) **Hashing the message:** For the message "ElGamal Cryptosystem", Bob gets the reduced integral hashing value of the message as follows:

- Compute its SHA-256 hash:

  `4ca0f4cc886bbbf987b930da47065c228fd93e5a441ceee5eecd1b72d9246fd7`.

- Convert the hash into an integer:
  $H_{hash} = 346601615629973149829732722879906350948638923162916116274742165030583399243735$.
- Reduce the above value modulo $\varphi(n) = 1610510$:

$$H \equiv H_{\text{hash}} (\text{mod } 1610510) \equiv 382265.$$

(b) **Signature of the message:**

- Bob chooses a random integer $k_s = 123457$ such that $\gcd(k_s, \varphi(n)) = 1$ and

$$k_s^{-1} (\text{mod } \varphi(n)) = 123457^{-1} (\text{mod } 1610510) \equiv 1387543.$$

- Compute S_1:
$$S_1 \equiv g^{k_s} (\text{mod } n) \equiv 2^{123457} (\text{mod } 1771561) \equiv 1024822.$$
- Compute S_2:
$$S_2 \equiv k_s^{-1} \cdot (H - b \cdot S_1) (\text{mod } \varphi(n)) \equiv 1387543 \cdot (-8841133) (\text{mod } 1610510) \equiv 556511.$$
- The signature pair, that is transmitted with the encrypted message, is given by
$$(S_1, S_2) = (1024822, 556511).$$

Phase IV. Decryption procedure:

Once Alice receives the ciphertext given in the pairs $(C_{1,i}, C_{2,i})$ with $1 \leq i \leq 20$ as recorded in Table 1, she uses her main private $a = 7$ to compute m_i:

$$m_i \equiv C_{2,i} \cdot (C_{1,i}^a)^{-1} (\text{mod } 1771561).$$

Then she recovers the plaintext from the decrypted ASCII codes m_i. The results of computations are summarized the following table:

Table 2: Decryption Results.

$C_{1,i}$	$C_{1,i}^a (\text{mod } n)$	$(C_{1,i}^a)^{-1} (\text{mod } n)$	$C_{2,i}$	m_i	Char
8	325591	826073	1207047	69	E
16	929745	1099839	1205044	108	l
4	16384	1215245	1163264	71	G
32	312773	1240381	222444	97	a
2	128	1425553	13952	109	m
8	325591	826073	1465790	97	a
16	929745	1099839	1205044	108	l
4	16384	1215245	524288	32	(space)
32	312773	1240381	1468620	67	C
2	128	1425553	14592	114	r
8	325591	826073	422169	121	y
16	929745	1099839	1380902	112	p
32	312773	1240381	850448	116	t
2	128	1425553	14208	111	o
8	325591	826073	240184	115	s
16	929745	1099839	890802	121	y
32	312773	1240381	537675	115	s
8	325591	826073	565775	116	t
4	16384	1215245	1654784	101	e

32	312773	1240381	432598	109	m

Phase V. Signature verification:

After obtaining the plaintext "ElGamal Cryptosystem" and receiving the signature pair $(S_1, S_2) = (1024822, 556511)$, Alice verifies the message as follows:

(a) **Hashing the message:** For the message "ElGamal Cryptosystem", Alice gets the reduced integral hashing value of the message in the same approach performed by Bob in *Phase III*.(*a*). Namely, she gets that

$$H \equiv H_{\text{hash}} (\text{mod } 1610510) \equiv 382265.$$

(b) **Verification Equation:** By using Bob's public key $r_2 = 512$, Alice verifies the validity of the signature by checking whether the following congruence holds:

$$g^H \equiv r_2^{S_1} \cdot S_1^{S_2} (\text{mod } n).$$

If the congruence holds, the signature is accepted as valid; otherwise, it is rejected. In fact, the left hand side is

$$\text{LHS} = g^H = 2^{382265} (\text{mod } 1771561) \equiv 302235.$$

On the other hand, the right hand side is

$$\text{RHS} = r_2^{S_1} \cdot S_1^{S_2} = 512^{1024822} \cdot 1024822^{556511} (\text{mod } 1771561) \equiv 302235.$$

Thus, the signature is valid.

3. Performance and Complexity Evaluation

In this section, we present a comprehensive comparison regarding the performance and complexity evaluation between our proposed cryptosystem (denoted by **OUR**), the classical ElGamal cryptosystem (denoted by **EGC**) by ElGamal [4] and a modification of ElGamal (denoted by **MEGC**) that is introduced by Mohit and Biswas [20] in which they modify the classical ElGamal cryptosystem with some change in the process of encryption and using a random session key for the encryption to give a more practical cryptosystem than the existing one. Furthermore, they modified the digital signature scheme of the ElGamal cryptosystem by changing the signature generation phase such as the calculation of two signatures.

3.1. Experimental Setup

For the comparison, we use the following parameters and tools:

- Prime modulus $p \approx 100$ bits and a generator $g \in U(p)$ and $U(p^2)$.
- A message size of 1000 characters, and each character random ephemeral keys $f_i \in [2, 15]$ are generated.
- All of the following results are computed using the SageMath program [24] using the same hardware and software setup. Note that, algorithms and data supporting the following findings are available on a reasonable request.

3.2. Encryption and Decryption Results

Table 3: Performance Comparison of ElGamal Cryptosystems.

Metric	**EGC**	**MEGC**	**OUR**
Prime Size	100-bit	100-bit	100-bit
Message Length (chars)	1000	1000	1000
Encryption Time (sec)	0.042	0.056	0.047
Decryption Time (sec)	0.018	0.029	0.019
Ciphertext Size (KB)	11	15	18
Signature Time (sec)	0.021	0.033	0.038

From the results of Table 3, it is clear that the classical ElGamal cryptosystem (EGC) is the fastest for encryption, decryption and signature scheme processes since it uses a simpler group $\mathbb{Z}_p^*$ comparing to the other cryptosystems. However, it offers the least ciphertext efficiency. After that, the OUR cryptosystem comes in the middle regarding the encryption and decryption timing, however it uses a very large multiplicative group U_p^t with $t > 2$. Regarding the ciphertext size and signature timing, OUR scheme is slower due to the use of the group $\mathbb{U}_{p^t}$, that produce a very secure cryptosystem as it enables both strong encryption and integrated signature with more robust algebraic structure.

3.3. Complexity Evaluation

For the computational complexity comparison, we assume that P, M and I denote the count of modular exponentiation, modular multiplication and modular inverse, respectively. The total count of the computations for each of the three cryptosystems is summarized in the following table:

Table 4: Computational Complexity Comparison.

Phase	**EGC**	**MEGC**	**OUR**
Key Generation	$2P$	$2P$	$4P$ (including MDHKEP)
Encryption	$2P + M$	$2P + M$	$3P + M$
Decryption	$P + I$	$P + I$	$P + I$
Signature Generation	$2P + M + I$	$2P + M + I$	$2P + M + I$
Signature Verification	$3P + M$	$3P + M$	$3P + M$

From Table 4, we see the additional $2P$ in the key generation of OUR scheme comes form using the MDHKEP to generate keys for determining the exponent t. Similarly, this extra key generation step over $U(p^t)$ leads to an additional P in the encryption process of OUR scheme. However, OUR scheme remains linear in complexity and similar to the other two cryptosystems in the decryption and signature procedures. Indeed, OUR scheme adds complexity for security purpose without major runtime penalties.

3.4. Ciphertext Expansion

In the following table, we present the results of comparing the three cryptosystems regarding to the ciphertext expansion factors for a fixed plaintext of size (1 KB).

Table 5: Ciphertext Expansion Rate.

Metric	EGC	MEGC	OUR
Plaintext Size (KB)	1	1	1
Ciphertext Size (KB)	11	15	18
Expansion Factor	11x	15x	18x

It is clear that OUR cryptosystem has a higher expansion factor comparing to the other two cryptosystems since it uses a larger modulus p^t with $t > 2$. However, this leads to larger ciphertext overhead, but it enables to enhance the security and joint support for encryption process and signing procedure. Note that, this result does not give how each cryptosystem scales with increasing the size of the plaintext. For that, see Figure 1 below.

3.5. Experimental Figures

Here, we give experimental figures regarding to the ciphertext expansion rate, encryption time and decryption time versus the increase of file size.

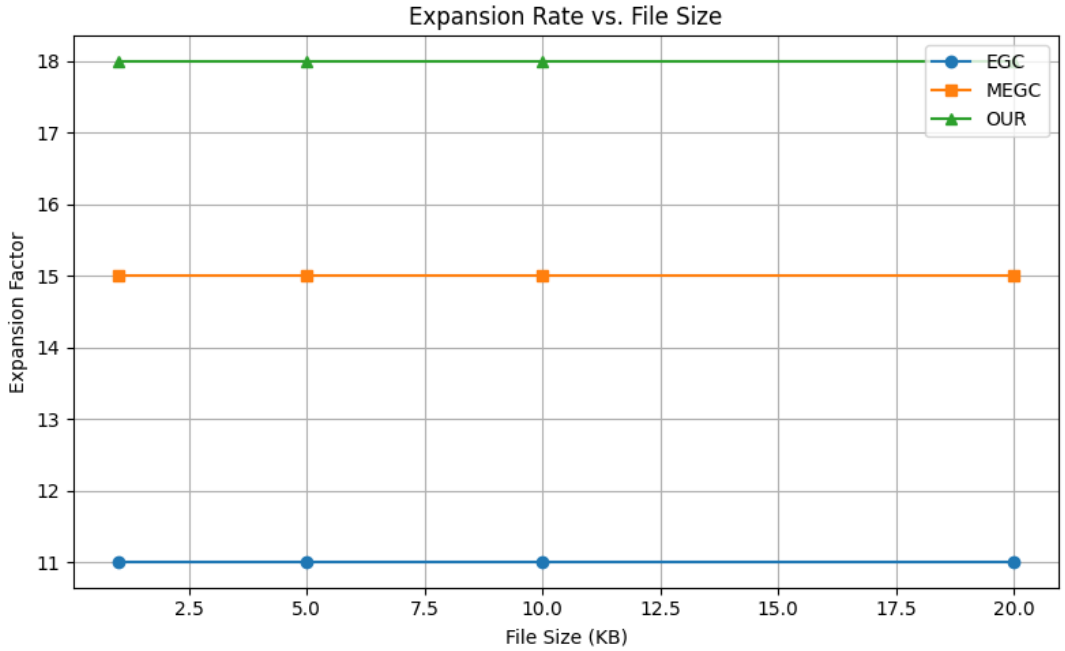

Figure 1: Expansion Rate vs. File Size for ElGamal Cryptosystems.

From Figure 1, we see that OUR cryptosystem has the highest expansion rate since it uses a very large prime power with the group of units $U(p^t)$, which reflects a cryptosystem that balances ciphertext size and security. However, other cryptosystems have low expansion rates, but at the cost of weaker integration.

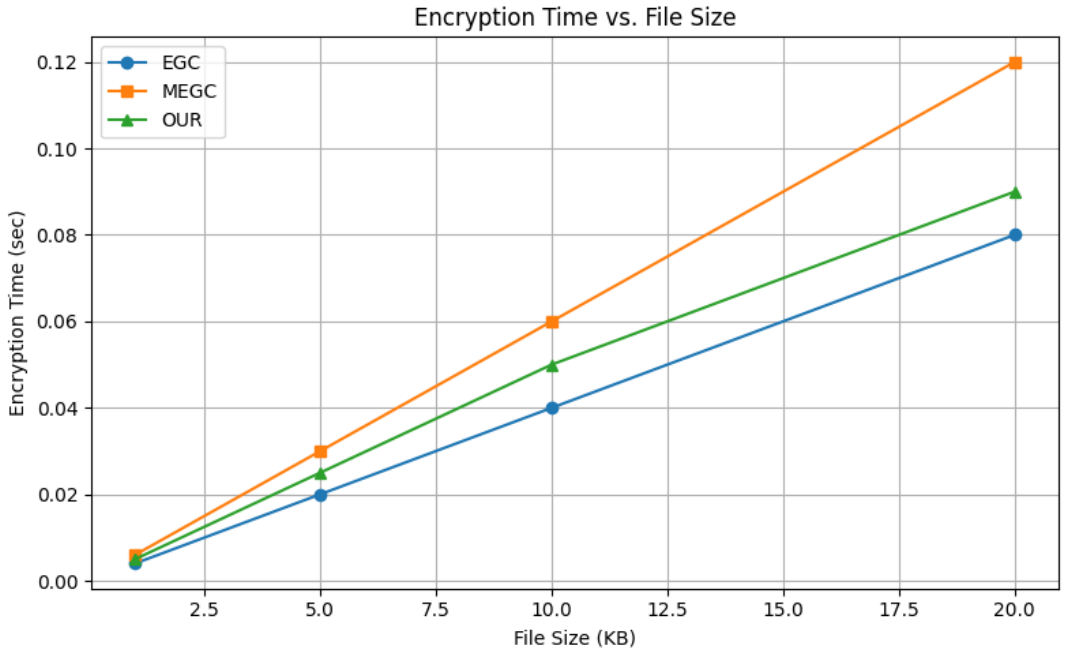

Figure 2: Encryption Time vs. File Size for ElGamal Cryptosystems.

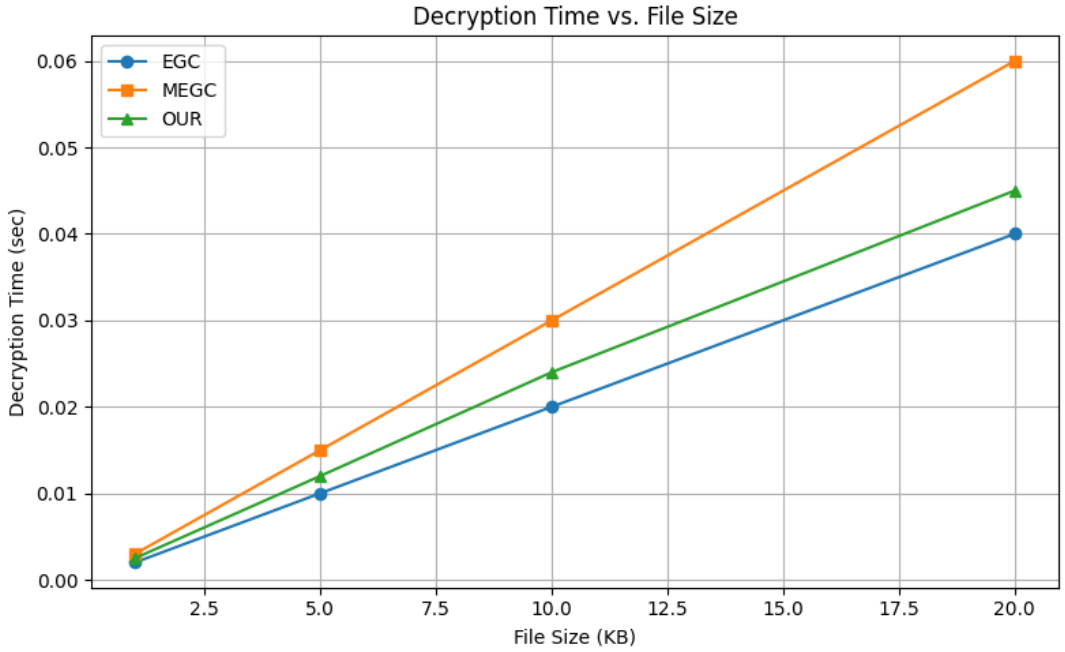

Figure 3: Decryption Time vs. File Size for ElGamal Cryptosystems.

From Figures 2 and 3 respectively, it is clear that EGC has the fastest encryption and decryption times due to its small modulus and minimal computation. After that comes OUR, that is faster than MEGC. However, OUR has a very large modulus p^t with $t > 2$ and more keys used in the encryption and decryption procedures (that lead to more security than the other cryptosystems), it still faster than the MEGC. Therefore, this supports OUR's suitability for scalable real-time encryption and indicates the successful optimization performance.

3.6. Security Comparison

The security of the three cryptosystems (**EGC**, **MEGC** and **OUR**) is based on the hardness of solving the discrete logarithm problem (DLP) over appropriately chosen cyclic groups, but they differ in structures,

computational domains and performances. In Table 6, we summarize the key aspects that evaluate and compare the security of these three cryptosystems:

Table 6: Security Comparison

Aspect	**EGC**	**MEGC**	**OUR**
Security Problem	DLP in $\mathbb{Z}_p^*$ or elliptic curve group	DLP with random session key in encryption	DLP in $U(p^t)$ combined with MDHKEP-derived ephemeral keys
Forward Secrecy	Not supported (fixed public key $y \equiv g^x \pmod p$)	Partially supported (session randomness in encryption)	Fully supported (the encryption and decryption exponents e, d are session-dependent by MDHKEP)
Key Reuse Attack Resistance	Vulnerable if keys are reused	Improved by enforcing session randomness	Strong, since every session regenerates fresh keys using DHKEP
Signature Security	Standard ElGamal signature, vulnerable to certain attacks if k_s is reused	More robust, modified signature with two components	Supports secure signing (with the hashing function SHA-256), resistant to reuse attacks due to session-derived exponents
Attack Surface	Attacks on DLP if the ephemeral f_i is weak	Same as EGC, but more resistant to ephemeral key reuse	Strengthened by the MDHKEP; secure under DLP assumption with larger group

In summary, **OUR** provides the strongest security among the three cryptosystems. Its security depends on the hardness of solving two discrete logarithm problems over the group of units $U(p^t)$: one is associated with the private key generation, and the other is with the MDHKEP-based ephemeral exponent t. This dual reliance enhances its long-term confidentiality and provides greater robustness against potential attacks compared to **EGC** and **MEGC**.

4. Conclusion and Future work

We presented a security enhancement of the classical ElGamal cryptosystem by using a large prime power modulus p^t where $t > 2$ is derived from a shared secret obtained by a modified Diffie-Hellman key exchange protocol. The system's security is also based on the DLP with much more complexity. Namely, this system has a richer algebraic structure, that increases the resistance to attacks. Furthermore, this cryptosystem enhances of the digital signature scheme of the classical ElGamal cryptosystem by incorporating the hash function (SHA-256) into the digital signature scheme, that gives more integrity for the signature of messages. Also, this cryptosystem is successfully implemented and performed using the SageMath software regarding to the performance of the cryptosystem's main phases given in Section 2. As a result, this cryptosystem is compared to the classical ElGamal cryptosystem (EGC) and another modification of the ElGamal cryptosystem (MEGC) regarding to the complexity evaluation (the results are given in 3.3), ciphertext expansion rate(the results are given in 3.4), and performance timing (the results are given in 3.5). The results of comparisons show that our cryptosystem is suitable for scalable real-time encryption applications as it is indicating successful performance optimization and robust integration of security features. For future

work, we will modify the mathematical structure of our cryptosystem for which it can be applied on image encryption.

Acknowledgments:

The author would like to thank the referees and the editor for their careful reading, useful comments and suggestions, which have significantly improved the quality of this paper.

References

[1] B. H. A. Alkfari and R. K. Ajeena, *The Huff curve ElGamal graphic public key cryptosystem*, J. Discrete Math. Sci. Cryptography **26:6** (2023), 1753–1760. 1

[2] D. S. Amirkhanova, M. Iavich and O. Mamyrbayev, *Lattice-based post-quantum public key encryption scheme using ElGamal's principles*, Cryptography **8:3** (2024). 1

[3] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Trans. Inf. Theory **22** (1976), 644–654. 1.4

[4] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, Advances in cryptology, Proc. CRYPTO '84, Univ. California, Santa Barbara, Aug. 19–22, 1984, Springer, Berlin (1985), 10–18. 1, 1, 3

[5] G. Fuchsbauer, A. Plouviez and Y. Seurin, *Blind Schnorr signatures and signed ElGamal encryption in the algebraic group model*, Advances in cryptology – EUROCRYPT 2020, Part II, Cham: Springer (2020), 63–95. 1

[6] H. H. Hadi and A. A. Neamah, *Diffie–Hellman key exchange based on block matrices combined with elliptic curves*, Int. J. Intell. Syst. Appl. Eng. **11:5s** (2023), 353–360. 1

[7] H. R. Hashim, *Curious properties of generalized Lucas numbers*, Bol. Soc. Mat. Mex., III. Ser. **27:3** (2021), Article 76. 1

[8] H. R. Hashim, *On the solutions of* $2^x + 2^y = z^2$ *in the Fibonacci and Lucas numbers*, J. Prime Res. Math. **19:1** (2023), 27–33. 1

[9] H. R. Hashim, A. Molnár and Sz. Tengely, *Cryptanalysis of ITRU*, Rad Hrvat. Akad. Znan. Umjet., Mat. Znan. **546** (2021), 181–193. 1

[10] P. Hecht, *Post-quantum cryptography: generalized ElGamal cipher over* $\mathrm{GF}(2^{518})$, Theor. Appl. Informatics **28:4** (2016), 1–14. 1

[11] Y. Ikematsu, S. Nakamura, B. Santoso and T. Yasuda, *Security analysis on an ElGamal-like multivariate encryption scheme based on isomorphism of polynomials*, Information security and cryptology, Inscrypt 2021, Cham: Springer (2021), 235–250. 1

[12] N. H. M. Ismail and M. Y. Misro, *Bézier coefficients matrix for ElGamal elliptic curve cryptosystem*, Malays. J. Math. Sci. **16:3** (2022), 483–499. 1

[13] B. J. Kadim and R. K. Ajeena, *Reliable public key cryptosystem type El-Gamal*, Proc. 2023 First Int. Conf. Advances in Electrical, Electronics and Computational Intelligence (ICAEECI) (2023), 1–6. 1

[14] A. W. Khaled and N. F. H. Al Saffar, *A survey on elliptic curves cryptosystems*, J. Adv. Res. Dyn. Control Syst. **11:1** (2019), 359–362. 1

[15] N. Koblitz, *Elliptic curve cryptosystems*, Math. Comput. **48** (1987), 203–209. 1

[16] N. Koblitz, A. Menezes and S. Vanstone, *The state of elliptic curve cryptography*, Des. Codes Cryptography **19:2–3** (2000), 173–193. 1

[17] A. K. Koppaka and V. N. Lakshmi, *ElGamal algorithm with hyperchaotic sequence to enhance security of cloud data*, Int. J. Pervasive Comput. Commun. **20:5** (2022), 607–619. 1

[18] N. Malyutina and V. Shcherbacov, *An analogue of the ElGamal scheme based on the Markovski algorithm*, ROMAI J. **17:1** (2021), 105–114. 1

[19] V. S. Miller, *Use of elliptic curves in cryptography*, Advances in cryptology – CRYPTO '85, Lect. Notes Comput. Sci. **218** (1986), 417–426. 1

[20] P. Mohit and G. P. Biswas, *Modification of ElGamal cryptosystem into data encryption and signature generation*, Proc. Int. Conf. Big Data, Machine Learning and Applications, Springer, Singapore (2021), 119–129. 3

[21] I. Niven, H. S. Zuckerman and H. L. Montgomery, *An introduction to the theory of numbers*, 5th ed., John Wiley & Sons, New York, 1991. 1

[22] R. Ranasinghe and P. Athukorala, *A generalization of the ElGamal public-key cryptosystem*, J. Discrete Math. Sci. Cryptography **25:8** (2022), 2395–2403. 1

[23] K. H. Rosen, *Elementary number theory and its applications*, 7th ed., Pearson, Boston, MA, 2023. 1

[24] W. A. Stein et al., *Sage Mathematics Software* (Version 9.0), The Sage Development Team, 2025, `http://www.sagemath.org`. 2.6, 3.1