The ISC Int'l Journal of Information Security
ISeCure

July 2023, Volume 15, Number 2 (pp. 216–229)
http://www.isecure-journal.org

Secure Outsourcing of Two Standard Identity-Based Cryptosystems **

Mohammad Reza Saeidi [1], and Hamid Mala [1,*]
[1] *Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran.*

ARTICLE INFO.

Article history:
Received: December 18, 2021
Revised: June 15, 2022
Accepted: April 30, 2023
Published Online: July 1, 2023

Keywords:
Checkability, Identity-Based Encryption, Outsourcing, Security

Type: Research Article

doi: 10.22042/isecure.2023.320345.734

dor: 20.1001.1.20082045.2023.15.2.2.1

ABSTRACT

In the last two decades, bilinear pairings have found many applications in cryptography. Meanwhile, identity-based cryptosystems based on bilinear pairings have received particular attention. The IEEE, IETF, and ISO organizations have been working on standardizing pairing-based cryptographic schemes. The Boneh-Franklin identity-based encryption and Sakai-Kasahara identity-based signature are the most well-known identity-based schemes that have been standardized. So far, researchers have proposed various schemes to reduce the computational overhead of pairing operations. All these schemes are trying to outsource pairing operations securely. However, besides pairing operations, there are other essential and costly operations in pairing-based cryptography and identity-based schemes, including scalar multiplication on elliptic curves. In this research, we outsource the Boneh-Franklin encryption in a more secure and efficient (in terms of computational and communication complexity) way than existing schemes. Also, we outsource the BLMQ signature (based on Sakai-Kasahara) scheme for the first time. The proposed schemes are secure in the OMTUP model. Also, unlike previous schemes, we considered communication channels insecure. Moreover, compared with the trivial solution, which outsources every single operation (such as pairing, scalar multiplication, and modular exponentiation) as a separate subroutine, our schemes offer less complexity by seamlessly outsourcing the whole encryption scheme for the first time.

1 Introduction

In recent years, cloud computing has attracted the attention of many researchers. Resource-constrained devices outsource their costly computing operations to cloud servers using cloud computing. In general, there are two entities in outsourcing computation. The first entity is a server with rich computing resources, and the second is a user with a resource-constrained device who, in a pay-per-use manner, delegates his heavy computation operations to the cloud server and

[*] Corresponding author.
**This paper is an extension of our previous work [1], published at the 18th ISCISC conference (ISCISC'2021). The most important difference between this paper and its conference version is the design of a scheme for secure outsourcing of the BLMQ signature.

Email addresses: mr_saeidi@eng.ui.ac.ir, h.mala@eng.ui.ac.ir

ISSN: 2008-2045 © 2020 ISC. All rights reserved.

thus benefits from unlimited computing resources. Despite its significant benefits, outsourcing computation has raised privacy concerns. Curious cloud servers should not be able to obtain valuable information about user input and computed output during outsourcing. Therefore, it is necessary to protect the privacy of sensitive user data. In addition, the cloud server may inadvertently or intentionally return invalid results. Therefore, users must be able to verify the results received from the servers, which is called *checkability*. The total computational cost of the client in all steps of the outsourcing protocol, including ensuring the data privacy and verification of the result, should be much lower than the original computation without outsourcing; otherwise, outsourcing is not reasonable. Regarding the number and trustability of servers, outsourcing schemes can be divided into three models: OUP, OMTUP, and TUP. In the OUP (One-Untrusted Program) model, the computations are outsourced to only one server, which can be malicious. In the OMTUP (One-Malicious version of a Two-Untrusted Program) model, the computations are outsourced to two untrusted servers, but only one may do malicious behavior. If we assume both servers can be malicious, the OMTUP model becomes TUP (Two-Untrusted Program) [2]. In the last two models, it is assumed that the servers do not collude. Servers can be considered as belonging to separate vendors. As a result, they lose their financial gain if the collusion of the servers is exposed.

Bilinear pairing is a heavy operation for resource-constrained devices such as IoT devices, which is used in many modern cryptographic protocols. Pairing-based cryptography is not only of interest to researchers, but also has commercial applications. Two prominent and leading companies in this field include Voltage Security and Trend Micro. Among the pairing-based schemes, identity-based encryption has received particular attention [3]. The idea of identity-based cryptosystems was first introduced in 1984 by Shamir. Using these systems, any pair of users can communicate securely and verify each other's signatures without exchanging public keys, keeping key directories, and using the services of a third party [4]. Any identity-based encryption system comprises four algorithms: setup, encryption, private key generation, and decryption, as first introduced in 2001 by Boneh and Franklin, hereafter denoted by BF. Moreover, the IEEE, IETF, and ISO have worked to standardize pairing-based cryptographic schemes. The BF encryption scheme is one of the encryption schemes specified in IEEE P1363.3 [5]. As both encryptors and decryptors must perform a pairing operation, this scheme's efficiency depends on the pairing operation's cost [3]. In addition, another costly operation in the BF encryption scheme is scalar multiplication. The only identity-based signature scheme mentioned in IEEE P1363.3 is BLMQ, which is based on the Sakai-Kasahara signature. BLMQ Scheme needs costly scalar multiplications in the signature generation phase and scalar multiplication and pairing operations in the verification phase.

1.1 Contributions

In this paper, for the first time, we outsource the BF encryption scheme in an integrated and secure manner to achieve more efficiency than separate outsourcing of its costly internal building blocks. Then, inspired by the outsourcing method of BF encryption, we also outsource the BLMQ signature for the first time. We present our schemes in the OMTUP model. Unlike previous schemes, we consider communication channels insecure. In addition, like the previous schemes offered in the OMTUP model, it assumes that the servers do not collude.

1.2 Paper Organization

This paper is organized as follows: Section 2 introduces some preliminaries. The related work is expressed in Section 3. The details of outsourcing BF encryption and BLMQ signature schemes are described in Section 4 and Section 5, respectively. Section 6 reviewed the proposed schemes regarding computational and communication complexity. Finally, the paper is concluded in Section 7.

2 Preliminaries

In this section, we first define bilinear pairing and scalar multiplication, and then describe the details of BF encryption and BLMQ signature schemes. Finally, we discuss the discrete logarithm problem.

2.1 Bilinear Pairings

Suppose G_1 and G_2 are both additive cyclic groups, G_T is a multiplicative cyclic group, and all three groups are of prime order p. Also, P is the generator of G_1 and Q is the generator of G_2. Then the function $e : G_1 \times G_2 \to G_T$ is a bilinear pairing if it has the following properties [6]:

- Bilinear: For all $A \in G_1$ and $B \in G_2$ we have $e(xA, yB) = e(A, B)^{xy}$.
- Non-degenerate: There exist $P \in G_1$ and $Q \in G_2$ such that $e(P, Q) \neq 1$, where 1 is the identity element in G_T.
- Computable: This means that $e(A, B)$ can be computed for all elements $A \in G_1$ and $B \in G_2$ in polynomial time.

2.2 Scalar Multiplication

Considering the integer $n \in \mathbb{N}^+$ and the point P on the elliptic curve E, the scalar multiplication nP on the elliptic curve E is defined as follows [7]:

$$nP = \underbrace{P + P + \cdots + P}_{n\ times} \tag{1}$$

2.3 BF Encryption Scheme According to IEEE P1363.3

Consider the four hash functions $H_1 : \{0,1\}^* \to G_1$, $H_2 : G_3 \to \{0,1\}^n$, $H_3 : \{0,1\}^n \times \{0,1\}^n \to Z_p^*$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$ and the n-bit random generator R_1. BF encryption scheme includes four phases as follows [5]:

(1) *Setup*: In this phase, a random number generator generates the parameters used in the other steps. The generation of the server secret, which must be kept secure, is done in this phase. All private keys will be calculated using this secret.
- Generate two additive cyclic groups, G_1, G_2, and a multiplicative cyclic group, G_3. All three are of prime order p and a pairing $e : G_1 \times G_2 \to G_3$.
- Select a random generator Q in G_2.
- Generate a random secret $s \in Z_p^*$ for the server.
- Calculate $R = sQ$.
- Announce the public parameter set $P = (R, Q, G_1, G_2, G_3, e)$.

(2) *Extract*: In this phase, the corresponding private key K_{ID} in G_1 is calculated using an arbitrary identity ID in $\{0,1\}^*$.
- Calculate the identity element as $M = H_1(ID)$.
- Calculate the private key corresponding to M as $K_{ID} = sM$ using the server secret s.

(3) *Encrypt*: The inputs of this phase are the arbitrary identity ID in $\{0,1\}^*$, the set of public parameters $P = (R, Q, G_1, G_2, G_3, e)$, and the message m of length n. Finally, the message m is encrypted, and the ciphertext E is generated.
- Calculate the random message o of length n using R_1.
- Calculate $M = H_1(ID)$.
- Calculate $r = H_3(o, m)$.
- Calculate $C_1 = rQ$.
- Calculate blinding value $B = e(rM, R)$.
- Calculate $C_2 = o \oplus H_2(B)$.
- Calculate $C_3 = m \oplus H_4(o)$.
- Create the cipthertext $E = (C_1, C_2, C_3)$ as output.

(4) *Decrypt*: In this phase, the ciphertext E and the receiver's private key (K_{ID}) are considered as input. Finally, the message m or *error* message appears in the output.
- Calculate the blinding value B using C_1 and K_{ID} as $B = e(K_{ID}, C_1)$.
- Calculate $o = C_2 \oplus H_2(B)$.
- Calculate $m = C_3 \oplus H_4(o)$.
- Calculate $r = H_3(o, m)$.
- Check that the equation $C_1 = rQ$ is valid—otherwise, output *error* and stop.
- Set the message m as output.

2.4 BLMQ Signature Scheme According to IEEE P1363.3

Consider the two hash functions $H_1 : \{0,1\}^* \to Z_p$ and $H_2 : \{0,1\}^* \times G_3 \to Z_p$. The BLMQ signature scheme includes four phases as follows [5]:

(1) *Setup*: In this phase, a random number generator generates the parameters used in the other steps. In addition, to calculate all private keys, the server secret must be generated in this phase and kept securely with the server.
- Generate two additive cyclic groups, G_1, G_2, and a multiplicative cyclic group, G_3. All three are of prime order p and a pairing $e : G_1 \times G_2 \to G_3$.
- Select a random generator Q_2 in G_2.
- Compute $Q_1 = \phi(Q_2)$ in G_1, where $\phi(Q_2)$ is a distortion map. Such a mapping maps a point $Q_2 \in E(GF(q))$ to a point $Q_1 \in E(GF(q^k))$ such that Q_1 and Q_2 are linearly independent.
- Generate a random secret $s \in Z_p^*$ for the server.
- Compute $R = sQ_2$.
- Compute $O = e(Q_1, Q_2)$.
- Publish the public parameter set $P = (R, O, Q_1, Q_2, G_1, G_2, G_3, e)$.

(2) *Extract*: In this phase, the corresponding private key K_{ID} in G_1 is calculated using an arbitrary identity ID in $\{0,1\}^*$.
- Calculate the identity element as $M = H_1(ID)$.
- Calculate the private key corresponding to M as $K_{ID} = (M + s)^{-1}Q_1$ using the server secret s.

(3) *Sign*: The inputs of this phase are the signer's private key K_{ID}, the set of public parameters $P = (R, O, Q_1, Q_2, G_1, G_2, G_3, e)$, the random integer $r \in Z_p^*$, and the message m. Finally, the signature (h, S) of the message m where $h \in Z_p$ and $S \in G_1$ is calculated.
- Calculate $u = e(Q_1, Q_2)^r$.
- Calculate $h = H_2(m, u)$.
- Calculate $S = (r + h)K_{ID}$.
- Output the signature as (h, S).

(4) *Verify signature*: In this phase, the signature

(h, S), signer's public key M and the set of public parameters $P = (R, O, Q_1, Q_2, G_1, G_2, G_3, e)$ are considered as input. If the claimed signature is accepted according to the public key M, the output is *valid* and otherwise *invalid*.

- Calculate $u = \frac{e(S, MQ_2+R)}{e(Q_1,Q_2)^h}$.
- If equation $h = H_2(m, u)$ is satisfied, the output is *valid* and otherwise *invalid*.

2.5 Discrete Logarithm Problem

Suppose p and q are prime numbers and $q \mid p-1$ and $g \in [1, p-1]$ is of order q. The private key is an integer x selected uniformly at random from the interval $[1, p-1]$, and the corresponding public key is $y = g^x \ mod\ p$. Determining the value of x by having the parameters (p, q, g) and the public key y is called the discrete logarithm problem [8].

3 Related Work

This section discusses the most important schemes for outsourcing bilinear pairing and scalar multiplication operations.

The paper by Chevallier-Mames *et al.* [9] can be considered the first work on bilinear pairings outsourcing. The schemes proposed in [9] are in the OUP model. The disadvantage of these schemes is that when outsourcing the pairing operation, the user has to compute other costly operations, such as scalar multiplication and modular exponentiation, locally.

In 2005, Kang *et al.* [10] improved the schemes presented in [9]. However, the user still had to perform costly operations such as scalar multiplication and modular exponentiation.

For the first time in 2007, Tsang *et al.* [6] discussed the batch pairing delegation and proved that batch outsourcing is significantly more efficient than multiple implementations of presented schemes in [9] and [10]. The checkability of all proposed schemes in [6, 9, 10] is 1. This means that the user can detect any error with probability 1.

Unlike previous works, Chen *et al.* [11] used two servers (OMTUP model) to outsource bilinear pairing operations with checkability $\frac{1}{2}$. Then, by using their scheme as a subroutine and another subroutine for scalar multiplication outsourcing [12], they outsourced the BF identity-based encryption and the Cha-Cheon identity-based signature. Liu and Cao [13] analyzed [11] by showing that incorrect results returned from the cloud servers are not always detectable. Then to solve this problem, unlike the OMTUP model, where one of the servers is malicious and the other is semi-honest, they considered both servers semi-honest.

Tian *et al.* [2] proposed two schemes for bilinear pairing outsourcing. The first scheme is in the OMTUP model, and its purpose is to increase efficiency, which in this regard has achieved less computational complexity than the scheme proposed in [11]. The second scheme is presented in the TUP model to increase checkability.

In [14], the authors presented two schemes for bilinear pairing outsourcing in the OMTUP model that have better computational, communication, and memory complexity than the similar schemes proposed until then. Also, both schemes reduce the computational complexity of the pre-computation phase.

The purpose of Luo *et al.* [15], as in [6], is to outsource a batch of pairing operations. The difference is that in [15], the product of n pairing is outsourced, while in [6], n distinct pairing operations are outsourced simultaneously.

In 2016, Lin *et al.* [16] proposed two schemes, *pair*, and *Fpair*, for outsourcing $e(A, B)$. The *pair* scheme is in the OMTUP model, and the *Fpair* scheme is in the TUP model. The online computational complexity of the user, the computational complexity of the servers, and the checkability of these schemes are similar to those presented in [2]. The advantage of schemes proposed in [16] over schemes offered in [2] is in the pre-computation phase.

In [17], Ren *et al.* proposed two schemes, the first for outsourcing $e(A, B)$ and the second for outsourcing $\prod_{i=1}^{n} e(A_i, B_i)$. In both schemes, two servers are used for outsourcing. The computational complexity of the first scheme is higher than in some previous schemes. However, the main advantage of the schemes proposed in [17] is their checkability so that in both schemes, the outsourcer detects any error with probability 1.

Tong *et al.* [18] proposed a scheme for bilinear pairing outsourcing without pre-computation in the OMTUP model with checkability 1. They first presented a scheme for secure outsourcing of scalar multiplication and then used it as a module for secure outsourcing of bilinear pairing. Therefore, the security of the proposed scheme for bilinear pairing outsourcing depends on the security of the proposed scheme for outsourcing scalar multiplication. Nevertheless, since the modulus of computations is considered secret in outsourcing scalar multiplication, this scheme is unsuitable for outsourcing BF encryption. In 2021, the authors in [19] enhanced their algorithm by applying the Ethereum blockchain to enable fair payments.

In 2019, Yang *et al.* [20] proposed two schemes, *Pai* and *SPai*, for multiple bilinear Pairings outsourcing. The *Pai* scheme outsources $e(A, B)$ and $e(C, D)$ simultaneously with checkability $\frac{1}{4}$, and the *SPai* scheme outsources $e(A, B)$ and $e(A, C)$ simultaneously with checkability $\frac{2}{7}$. Both schemes are presented in the OUP model.

In [21], the authors presented a scheme for outsourcing bilinear pairing to a few servers (from 1 to n). Servers can be malicious or semi-honest. The difference between their scheme and previous schemes is the resistance to collusion of servers. Even if all servers collude, the user can detect forged computation results with high probability. For this reason, the computational complexity of the scheme presented in [21] is greater than the schemes that do not consider collusion of servers.

Lin *et al.* [22] proposed the first blockchain-based design for computation outsourcing in 2020. They adapted the scheme presented in [11] to form a blockchain-based system to secure outsourcing bilinear pairings. The authors presented their scheme in the OUP model with checkability $\frac{3}{4}$ and replaced permissioned nodes with a trusted server to generate random tuples. To ensure scalability, they designed a smart contract to build the exchange relationship between a user and a computation provider.

One of the first independent papers on scalar multiplication outsourcing is [23], which was presented in 2016 by Zhou and Ren. The goal is to securely outsource sP operation in the OUP model, where s is an integer and P is a point on an elliptic curve. To this purpose, the double and add algorithm in projective coordinates have been used to eliminate inverse operations. The main idea is to map the field to a ring homomorphism, then calculate the scalar multiplication by the server in the ring. Finally, the user recovers the main result in the field from the result obtained in the ring. The authors of [23] in another paper [24], in addition to presenting a scheme for the outsourcing of modular exponentiation using the previous idea, added the checkability feature to the scalar multiplication outsourcing scheme presented in [23]. The modulus of computations in the mentioned schemes is considered secret. Therefore, these schemes are not suitable for outsourcing BF encryption.

In 2014, Chen *et al.* [12] first proposed a scheme for secure outsourcing of modular exponentiation and then extended it to outsourcing scalar multiplication operations. The schemes presented in [12] are in the OMTUP model.

In 2020, Ping *et al.* [25] proposed two schemes, first for outsourcing modular inversion using a single server and second for outsourcing scalar multiplication operation using two non-colluding servers. The checkability of the first and second schemes are 1 and $\frac{3}{4}$, respectively. Also, they used a one-time-pad encryption model to preserve the privacy of the input and output of their schemes.

In addition to pairing and scalar multiplication, outsourcing other costly operations interests researchers, some of which are discussed below. In 2022, Zhao *et al.* [26] proposed a secure outsourcing algorithm that allows devices to outsource the shortest path computing tasks to a single cloud server with checkability close to 1. Today, privacy-preserving message transmission and authentication have become indispensable parts of data collection and analysis. Hence Li *et al.* [27] designed a privacy-preserving and verifiable outsourcing message transmission and authentication protocol, allowing the resource-constrained users to delegate complex operations to the two untrusted servers. Matrix inversion is a widely used but costly operation for IoT devices in edge computing. In 2022, Gao *et al.* [28] proposed two parallel privacy-preserving outsourcing schemes for matrix inversion in IoT to solve this problem. They outsourced their schemes to two edge servers.

4 Secure and Efficient Outsourcing of BF Encryption Scheme

In this section, we introduce the pre-computation phase and its implementation methods. Then to reduce the online computation phase, we present the BF encryption outsourcing scheme called *BFEnc_O* and the BF decryption outsourcing scheme called *BFDec_O*.

4.1 Pre-Computation

In bilinear pairing outsourcing schemes, a function usually denoted as *Rand* is used as a pre-computation to speed up the client's online computations. The *Rand* function generates random output upon receiving the appropriate input. [29] introduces two methods for implementing the *Rand* function. In the first method, the construction of a table of random values is left to a trusted server. Each table row is the output of a single *Rand* function call. In the second method, the *Rand* function holds two data tables. The first table is called the static table (ST), and the second table is called the dynamic table (DT). Whenever the *Rand* function is invoked, a row from the DT is returned and cleared. Dynamic tables are based on static tables, and the client itself creates both tables. The client beforehand makes a static table and generates a dynamic table from it. As mentioned, each row of the dynamic table returned as output is deleted, but when the client is idle, it creates new rows again and fills in the dynamic table using new random values. In the following schemes, it is assumed that the pre-computation phase is implemented according to the second method. If the first method is used, the lookup table is not loaded on the client memory through the network to eliminate the possibility of eavesdropping. Therefore, the random values generated in the pre-computation phase in our schemes, similar to many schemes mentioned in the related work section [2, 6, 11, 12, 14–17, 20, 21], remain confidential in the user side. It should be noted that the disclosure of random values generated in the pre-computation phase (except those disclosed to untrusted servers during outsourcing) pose a security risk. In addition, values that can be generated offline are generated in the pre-computation phase. For example, we will outsource the scalar multiplications rQ, while the scalar multiplication xQ will be generated offline in the pre-computation phase. This is because in BF's original scheme, $r = H_3(o, m)$, where m is a message sent or received by the user and generated online. Therefore, it is impossible to generate rQ offline.

4.2 BF Encryption Outsourcing Scheme

The $BFEnc\_O$ is in the OMTUP model. To secure communication between the client and the servers, the client already agrees with the first server on the symmetric key k_1 and the second server on the symmetric key k_2. Then, if necessary, the messages will be encrypted. We denote the encryption of a message using the key $k_{i,i\backslash in\{1,2\}}$ by the symbol E_{k_i}. We also use the $Rand_{BFEnc}$ function as the pre-computation phase to speed up online computations. Figure 1 shows the input and output of the $BFEnc\_O$ scheme and the $Rand_{BFEnc}$ function. The details of the $BFEnc\_O$ scheme are explained in Figure 2.

The idea behind the $BFEnc\_O$ scheme is to outsource costly operations according to their features. For example, in outsourcing scalar multiplications rQ and rM, there is no need to preserve the privacy of the Q and M points because Q is the public parameter of the system, and M is the identifier of the recipient of the message. Also, in outsourcing $e(rM, R)$, there is no need to preserve the privacy of R as one of the public parameters. For this reason, our scheme is more efficient than the one presented in [11]. In outsourcing rQ and $e(rM, R)$, we obtained checkability 1 by sending similar requests to two servers and checking the equality of the received responses because, in the OMTUP model, only one of the servers may behave maliciously. Nevertheless, the total checkability is limited to outsourcing rM. Because $\sigma + 1$ requests are sent to the servers, each server with a probability of $\frac{1}{\sigma+1}$ can distinguish the request which is not for testing and consequently can return a wrong value. As a result, the user checkability of outsourcing rM is $1 - \frac{1}{\sigma+1} = \frac{\sigma}{\sigma+1}$. Therefore, the total checkability of the $BFEnc\_O$ scheme is $\frac{\sigma}{\sigma+1}$.

***BFEnc_ O* scheme input:**

An arbitrary identity $ID \in \{0,1\}^*$.

Public parameter set $P = (R, Q, G_1, G_2, G_3, e)$.

n-bit message m.

***BFEnc_ O* scheme output:**

Ciphertext E or $error$.

***Rand*$_{BFEnc}$ function input:**

Public parameter set $P = (R, Q, G_1, G_2, G_3, e)$.

Keys k_1 and k_2.

Parameter σ (related to checkability).

***Rand*$_{BFEnc}$ function output:**

$(x, xQ, W, e(-W, R)^{-1}, r', E_{k_1}(r'), E_{k_1}(r_1), \cdots,$

$E_{k_1}(r_\sigma), E_{k_2}(r_1), \cdots, E_{k_2}(r_\sigma)),$

where: $x, r', r_1, \cdots, r_\sigma \in_R Z_p^*$, $W \in_R G_1, xQ \in_R$

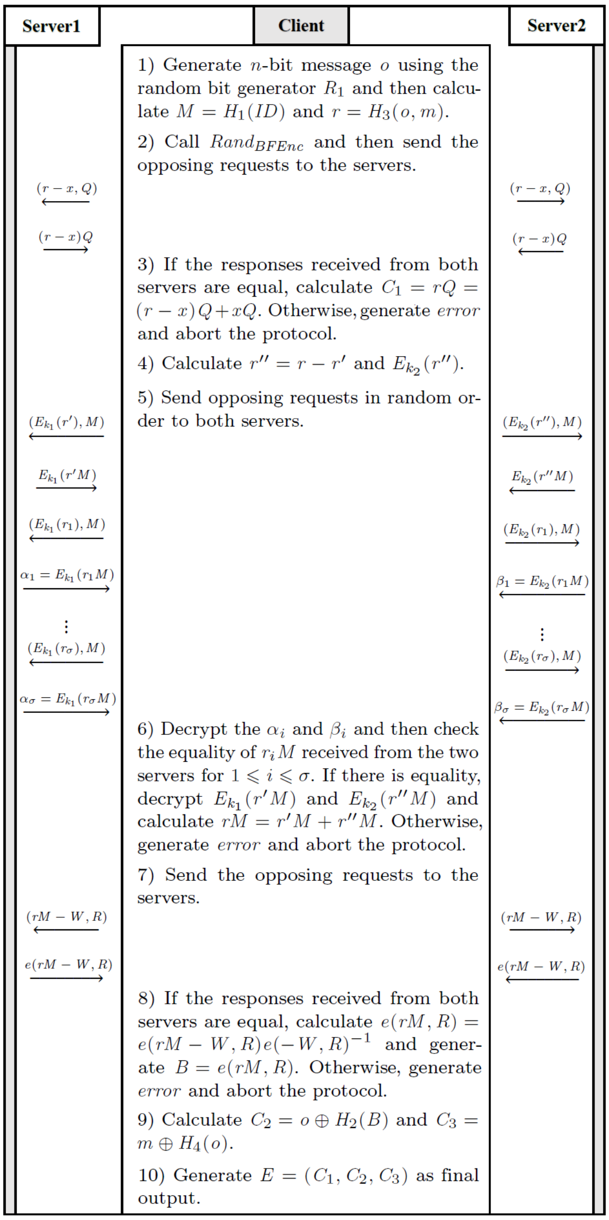
$G_2, e(-W, R)^{-1} \in G_3$.

Figure 1. Input and output of the $BFEnc\_O$ scheme and the $Rand_{BFEnc}$ function

4.3 BF Decryption Outsourcing Scheme

Similar to the previous scheme, the $BFDec\_O$ scheme is also in the OMTUP model. Here also, to secure communication between the client and the servers, the client already agrees with the first server on the symmetric key k'_1 and the second server on the symmetric key k'_2. Then, if necessary, the messages will be encrypted. We also use the $Rand_{BFDec}$ function as the pre-computation phase to speed up online computations. Figure 3 shows the input and output of the $BFDec\_O$ scheme and the $Rand_{BFDec}$ function, and the details of the $BFDec\_O$ scheme are explained in Figure 4.

Here we have outsourced rQ similar to the $BFEnc\_O$ scheme. Therefore, the checkability of outsourcing rQ in the $BFDec\_O$ scheme equals 1. To outsource $e(K_{ID}, C_1)$, the client must outsource a bilinear pairing operation and an exponentiation operation. But the computational complexity of client and servers for outsourcing $e(K_{ID}, C_1)$ in the $BFDec\_O$ scheme is less than their computational complexity in [11] because, in outsourcing $e(K_{ID}, C_1)$, there is no need to preserve the privacy of C_1. On the other hand, because $e(K_{ID}, C_1)$ is involved in the retrieval of r by the client (decryptor), in the $BFDec\_O$ scheme, the checkability of outsourcing $e(K_{ID}, C_1)$ depends on the checkability of outsourcing rQ. Since the client



1) Generate n-bit message o using the random bit generator R_1 and then calculate $M = H_1(ID)$ and $r = H_3(o,m)$.

2) Call $Rand_{BFEnc}$ and then send the opposing requests to the servers.

3) If the responses received from both servers are equal, calculate $C_1 = rQ = (r-x)Q + xQ$. Otherwise, generate *error* and abort the protocol.

4) Calculate $r'' = r - r'$ and $E_{k_2}(r'')$.

5) Send opposing requests in random order to both servers.

6) Decrypt the α_i and β_i and then check the equality of r_iM received from the two servers for $1 \leqslant i \leqslant \sigma$. If there is equality, decrypt $E_{k_1}(r'M)$ and $E_{k_2}(r''M)$ and calculate $rM = r'M + r''M$. Otherwise, generate *error* and abort the protocol.

7) Send the opposing requests to the servers.

8) If the responses received from both servers are equal, calculate $e(rM,R) = e(rM-W,R)e(-W,R)^{-1}$ and generate $B = e(rM,R)$. Otherwise, generate *error* and abort the protocol.

9) Calculate $C_2 = o \oplus H_2(B)$ and $C_3 = m \oplus H_4(o)$.

10) Generate $E = (C_1, C_2, C_3)$ as final output.

Figure 2. *BFEnc_O* scheme

outsources rQ with checkability 1 and then checks the equation $C_1 = rQ$ if one of the servers is malicious in the outsourcing of $e(K_{ID}, C_1)$ in step 4 of Figure 4, the equation $C_1 = rQ$ is not established in step 8, and the client aborts the protocol. In other words, we used the decryption phase feature of BF's original scheme to ensure the validity of the responses received from the servers in step 4 of the *BFDec_O* scheme. As a result, the total checkability of the *BFDec_O* scheme is 1. The computational complexity of the *BFDec_O* scheme is less than that of the scheme presented in [11] due to the proportionality of the outsourcing of each costly operation with its features (private/public and fixed/variable scalar coefficients and points).

***BFDec_ O* scheme input:**

Ciphertext $E = (C_1, C_2, C_3)$.

Recipient's private key (K_{ID}).

Public parameter set $P = (R, Q, G_1, G_2, G_3, e)$.

***BFDec_ O* scheme output:**

Plaintext m or *error*.

***Rand*$_{BFDec}$ function input:**

Public parameter set $P = (R, Q, G_1, G_2, G_3, e)$.

Keys k'_1 and k'_2.

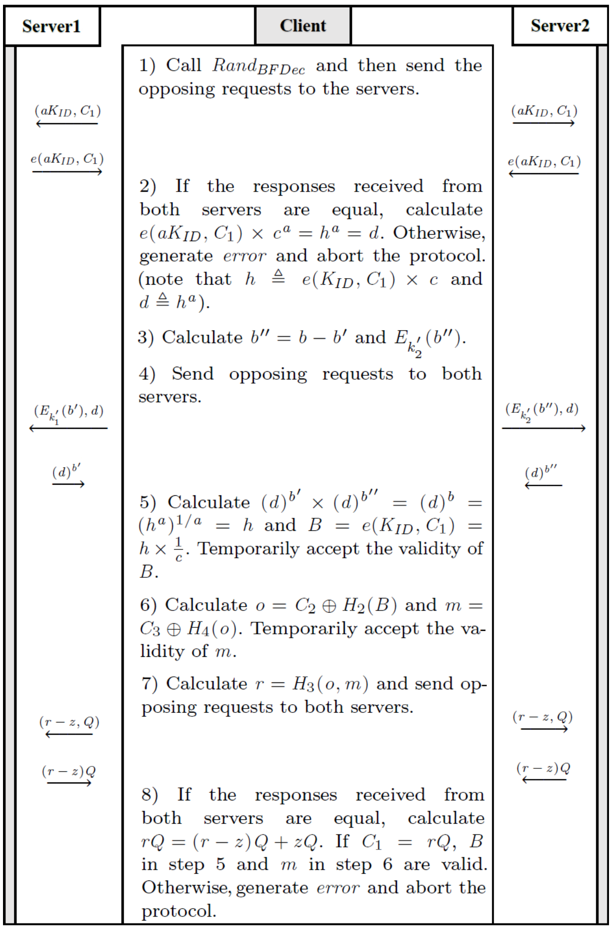
Recipient's private key (K_{ID}).

***Rand*$_{BFDec}$ function output:**

$(a, aK_{ID}, b = \frac{1}{a}, c, \frac{1}{c}, c^a, b', E_{k'_1}(b'), z, zQ)$,

where: $a, b', z \in_R Z_p^*, aK_{ID} \in_R G_1, zQ \in_R G_2, c \in_R G_3$.

Figure 3. Input and output of the *BFDec_O* scheme and the $Rand_{BFDec}$ function



1) Call $Rand_{BFDec}$ and then send the opposing requests to the servers.

2) If the responses received from both servers are equal, calculate $e(aK_{ID}, C_1) \times c^a = h^a = d$. Otherwise, generate *error* and abort the protocol. (note that $h \triangleq e(K_{ID}, C_1) \times c$ and $d \triangleq h^a$).

3) Calculate $b'' = b - b'$ and $E_{k'_2}(b'')$.

4) Send opposing requests to both servers.

5) Calculate $(d)^{b'} \times (d)^{b''} = (d)^b = (h^a)^{1/a} = h$ and $B = e(K_{ID}, C_1) = h \times \frac{1}{c}$. Temporarily accept the validity of B.

6) Calculate $o = C_2 \oplus H_2(B)$ and $m = C_3 \oplus H_4(o)$. Temporarily accept the validity of m.

7) Calculate $r = H_3(o,m)$ and send opposing requests to both servers.

8) If the responses received from both servers are equal, calculate $rQ = (r-z)Q + zQ$. If $C_1 = rQ$, B in step 5 and m in step 6 are valid. Otherwise, generate *error* and abort the protocol.

Figure 4. *BFDec_O* scheme

5 Secure Outsourcing of BLMQ Signature Scheme

This section describes the BLMQ signature generation outsourcing scheme called *BLMQSign_O* and the BLMQ signature verification outsourcing scheme called *BLMQVerify_O*. It should be noted that the *BLMQSign_O* scheme requires a pre-computation phase similar to that described in Section 4.

5.1 BLMQ Signature Generation Outsourcing Scheme

The *BLMQSign_O* scheme is also, like the other schemes presented in this paper, in the OMTUP model. This scheme does not need to share the symmetric keys between the client and the servers because only the masking technique is used. We use the $Rand_{BLMQSign}$ function to speed up the client's online computations during outsourcing. When the client invokes this function, it returns four random values independent of the previous. The inputs and outputs of the *BLMQSign_O* scheme and the $Rand_{BLMQSign}$ function are shown in Figure 5, and the details of the *BLMQSign_O* scheme are shown in Figure 6.

The first step in the BLMQ signature generation is to calculate $u = O^r = e(Q_1, Q_2)^r$, where r is randomly selected from Z_p^*, and $e(Q_1, Q_2)$ is the public parameter of the system. Therefore, there is no need to outsource u, which can be generated offline in the pre-computation phase. The next step is to calculate $h = H_2(m, u)$, and since the message m is generated online, h must also be calculated online. The user only needs to outsource the scalar multiplication $(r + h)K_{ID}$. In outsourcing $S = (r + h)K_{ID}$, the values $r + h$ and K_{ID} are masked ($\frac{r+h}{b}$ and bK_{ID}, respectively) and sent to the servers. Due to the equality check of responses by the client, the checkability of outsourcing $S = (r + h)K_{ID}$ and consequently the checkability of the *BLMQSign_O* scheme is 1. The scalar multiplication $(r + h)K_{ID}$ can be outsourced in another way. So that rK_{ID} is generated in the pre-computation phase, and the calculation of hK_{ID} is outsourced. Then, the user can reach the $(r + h)K_{ID}$ with a point addition operation. It is clear that in the second method, the user's computational complexity is higher due to the existence of the point addition operation. Also, in the case of outsourcing point addition $rK_{ID} + hK_{ID}$, the user communication overhead will increase.

***BLMQSign_ O* scheme input:**

Signer's private key (K_{ID}).

Public parameter set $P = (R, O, Q_1, Q_2, G_1, G_2, G_3, e)$.

A message m to be signed.

***BLMQSign_ O* scheme output:**

Signature (h, S) or $error$.

$Rand_{BLMQSign}$ function input:

Public parameter set $P = (R, O, Q_1, Q_2, G_1, G_2, G_3, e)$.

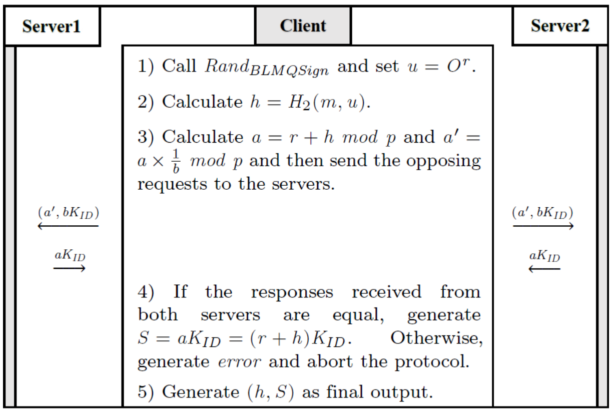
Signer's private key (K_{ID}).

$Rand_{BLMQSign}$ function output:

$(r, O^r = e(Q_1, Q_2)^r, \frac{1}{b}, bK_{ID})$,

where: $r, b \in_R Z_p^*, bK_{ID} \in_R G_1, e(Q_1, Q_2)^r \in_R G_3$.

Figure 5. Input and output of the *BLMQSign_O* scheme and the $Rand_{BLMQSign}$ function



1) Call $Rand_{BLMQSign}$ and set $u = O^r$.

2) Calculate $h = H_2(m, u)$.

3) Calculate $a = r + h \ mod \ p$ and $a' = a \times \frac{1}{b} \ mod \ p$ and then send the opposing requests to the servers.

4) If the responses received from both servers are equal, generate $S = aK_{ID} = (r + h)K_{ID}$. Otherwise, generate $error$ and abort the protocol.

5) Generate (h, S) as final output.

Figure 6. *BLMQSign_O* scheme

5.2 BLMQ Signature Verification Outsourcing Scheme

The *BLMQVerify_O* scheme is also in the OMTUP model. Since anyone can perform the verification phase without any secret parameter, we do not need to mask the values or create a secure channel in this scheme. Therefore, it is only necessary to check the correctness of the servers' responses. To do this, the client outsources the BLMQ signature verification phase with a checkability of 1 by sending similar requests to two servers and comparing the received responses. The input and output of the *BLMQVerify_O* scheme and its details are shown in Figure 7 and Figure 8, respectively. According to Figure 8, for outsourcing the BLMQ signature verification, the client must outsource three operations scalar multiplication, pairing, and modular exponentiation. This method is used when the servers perform only the mentioned three operations. However, suppose we assume that the servers perform the entire signature verification operation. In this case, the client can interact with each server independently and send parameters $\{M, Q_2, S, R, O, h\}$ to them through just one connection and then receive u as a response. In the next step, the client checks the

***BLMQVerify_ O* scheme input:**

The public key element (M).

Public parameter set $P = (R, O, Q_1, Q_2, G_1, G_2, G_3, e)$.

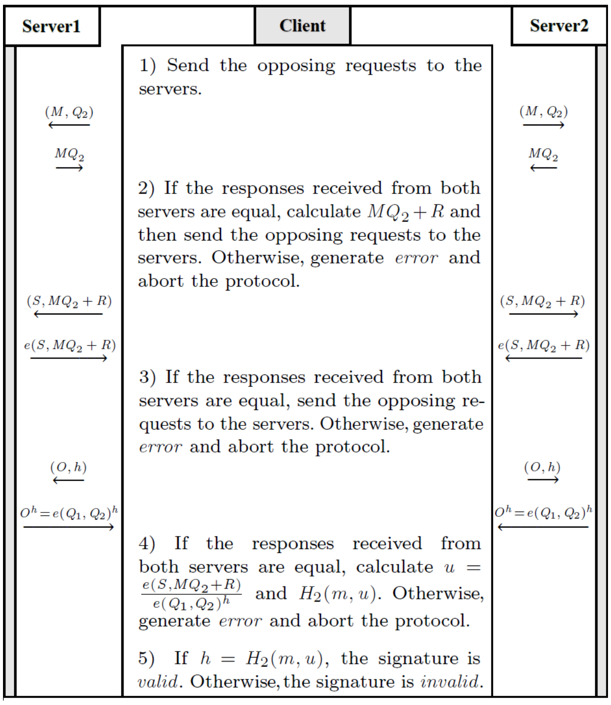
Message m.

Signature (h, S).

***BLMQVerify_ O* scheme output:**

Valid or *invalid* or *error*.

Figure 7. Input and output of the *BLMQVerify_O* scheme



1) Send the opposing requests to the servers.

2) If the responses received from both servers are equal, calculate $MQ_2 + R$ and then send the opposing requests to the servers. Otherwise, generate *error* and abort the protocol.

3) If the responses received from both servers are equal, send the opposing requests to the servers. Otherwise, generate *error* and abort the protocol.

4) If the responses received from both servers are equal, calculate $u = \frac{e(S, MQ_2+R)}{e(Q_1, Q_2)^h}$ and $H_2(m, u)$. Otherwise, generate *error* and abort the protocol.

5) If $h = H_2(m, u)$, the signature is *valid*. Otherwise, the signature is *invalid*.

Figure 8. *BLMQVerify_O* scheme

equality of the responses received from both servers. It is clear that in the second method, the user's computational and communication complexity is less.

6 Evaluation

In this section, we first discuss the security of the proposed schemes and then evaluate their computational and communication complexity.

6.1 Security of BF Encryption Outsourcing Scheme

According to Figure 2, in the *BFEnc_O* scheme, two scalar multiplication operations and one bilinear pairing operation are outsourced. We explain the security of these three operations below. Other operations are calculated locally by the client.

- *Outsourcing* rQ: In this outsourcing, the privacy of r and rQ must be maintained, but Q is the public parameter of the system. Instead of r, $r - x$, is sent to the servers. Since x is a random element with a uniform distribution of Z_p^* generated in the pre-computation phase, by knowing $r - x$ the distribution of r is still uniform. Hence servers or eavesdroppers cannot reach r from $r - x$. On the other hand, the servers return $(r - x)Q$ in response, which makes it impossible to reach rQ by knowing $(r - x)Q$, because only the client knows the values of r, x, and xQ.
- *Outsourcing* rM: In this outsourcing, the privacy of r and rM must be maintained, but M is the identifier of the recipient and therefore disclosed. r is decomposed into two random values r' and r'' as $r'' = r - r'$ and r' is a random value generated in the pre-computation phase. The values r' and r'' are encrypted using the client's shared symmetric key with each server, and only one of them is sent to each server. As a result, decrypting r' and r'' by the eavesdropper and reaching r depends on the security of the AES encryption algorithm. On the other hand, due to the assumption that the servers do not collude, they cannot reach r. Also, only the client with its symmetric key can access $r'M$ and $r''M$ and then calculate $rM = r'M + r''M$.
- *Outsourcing* $e(rM, R)$: The point R is a public parameter of the system, but rM and $e(rM, R)$ must be kept private. The client masks the rM using the random point W as $rM - W$. On the other hand, the client multiplies the response received from the servers by $e(-W, R)^{-1}$(generated in the pre-computation phase) to reach the $e(rM, R)$. No entity other than the client can compute $e(-W, R)^{-1}$, because it does not have the point W.

According to Figure 4, in the *BFDec_O* scheme, pairing operation $e(K_{ID}, C_1)$ and scalar multiplication rQ are outsourced and other operations are calculated locally by the client. The security of rQ is proven as before. But we will explain the security of outsourcing $e(K_{ID}, C_1)$ below.

- *Outsourcing* $e(K_{ID}, C_1)$: K_{ID} is the client's private key and C_1 is a part of the ciphertext. Therefore, the privacy of K_{ID} and $e(K_{ID}, C_1)$ must be preserved. The client first sends (aK_{ID}, C_1) to the servers, where the random value a and the scalar multiplication aK_{ID} are both generated in the pre-computation phase. Note that servers and eavesdroppers cannot reach private key K_{ID} by knowing aK_{ID}, because both a and K_{ID} are unknown. The client then receives $e(aK_{ID}, C_1)$ from the servers and multiplies it by c^a to obtain d (note that c and a are only available to the client). The client then subtracts b (inverse of a) from the random value b' (generated in the pre-computation phase) to reach b''. In the next step, the client encrypts b' and b'' using the symmetric keys and sends $(E_{k'_1}(b'), d)$ and $(E_{k'_2}(b''), d)$ to the first and second server, respectively. Neither the servers nor the eavesdropper can achieve the inverse of a, because they do not have access to both b' and b''. The client sends d to the servers without encryption. In response to the client, the first server calculates and returns $(d)^{b'}$ and the second server $(d)^{b''}$. It should be noted that the eavesdropper needs to solve the discrete logarithm problem to obtain b' and b'' from $(d)^{b'}$ and $(d)^{b''}$, respectively. The client obtains $e(K_{ID}, C_1) \times c$ by calculating $(d)^{b'} \times (d)^{b''}$, and finally reaches $e(K_{ID}, C_1)$ by a simple division. Note that no entity other than the client can access $e(K_{ID}, C_1)$ due to lack of c.

6.2 Security of BLMQ Signature Outsourcing Scheme

According to Figure 6, in the *BLMQSign_O* scheme, scalar multiplication operation $(r + h)K_{ID}$ is outsourced. We explain the security of this operation below. Other operations are calculated locally by the client.

- *Outsourcing* $(r + h)K_{ID}$: r is a random number selected by the signer in the first step of the signature generation phase and its privacy must be preserved. The privacy of the signer's private key K_{ID} must also be preserved. However, there is no need to preserve the privacy of h and $(r + h)K_{ID}$ because, as the output of the signature generation phase, they are sent explicitly to the recipient. The signer has the random value b and the bK_{ID} from the pre-computation phase. The signer first divides $r + h$ by b and then sends the pair $(\frac{r+h}{b}, bK_{ID})$ to both servers. Obviously, achieving r from $\frac{r+h}{b}$ is impossible for servers and eavesdroppers because b is not available. It is also not possible to get K_{ID} or b from bK_{ID}.

According to Figure 8, in the *BLMQVerify_O* scheme, the client outsources operations MQ_2, $e(S, MQ_2 + R)$, and $e(Q_1, Q_2)^h$. Given that all values are public (M is the signer identifier, R, Q_2, and $e(Q_1, Q_2)$ are the public parameters of the system, and (S, h) is the signature generated by the signer), there is no need for privacy at any step of outsourcing. Other operations are performed locally by the client.

6.3 Computational Complexity

First, we use [30] and [31] to unify the units of operations. We consider elliptic curves of the supersingular type with 80-bit security. The length of elements of groups Z_p, G_1, G_2, and G_3 in a supersingular curve with 80-bit security is 160, 512, 512, and 1024 bits, respectively [30]. Table 1 shows the ratio of times of different operations.

Table 1. Estimation of calculation time of different operations on a supersingular curve with 80-bit security

The ratio of times compared to fixed-base exponentiation		
G_1	Fixed-base exponentiation	1
	General exponentiation	5
G_2	Fixed-base exponentiation	1
	General exponentiation	5
	Hashing	5
G_3	Fixed-base exponentiation	1
	General exponentiation	5
	Pairing	50

It should be noted that we ignore the computational complexity of modular addition, XOR, and hashing to a string or into Z_p, as opposed to modular multiplication, point addition, scalar multiplication, fixed-base, and general exponentiation, hashing into G_1 or G_2, bilinear pairing and AES encryption/decryption. There is a hashing into G_1 in the *BFEnc_O* scheme $(M = H_1(ID))$ that the client can calculate once and use many times or outsource without privacy concerns. Therefore, we ignore its computational complexity.

In [23], the cost of point addition and doubling is calculated as 14 and 12 modular multiplications, respectively. Also, the cost of scalar multiplication sP is estimated to be equivalent to the cost of $12u + 14v$ modular multiplications in which $u = \log_2 s$ and v is equal to the number of 1's in the binary representation of s and $0 < v < u$. According to previous explanations, the scalar multiplication coefficients in our schemes are 160 bits. Therefore $u = 160$ and $0 < v < 160$. If we consider v by an average of 80 bits, each scalar multiplication, approximately, equals 3040 modular multiplications.

Since AES works over 128-bit blocks, we need 2, 4, and 8 AES blocks to encrypt the elements of Z_p, G_1, or G_2 and G_3, respectively. Also it should be noted that to encrypt a point of G_1 or G_2, it is enough to encrypt just one of its coordinates x or y. The cost of encrypting each 128-bit AES block is 256 cycles [31]. We assume that these 256 cycles are the result of the T-table implementation of AES on a 32-bit processor [32]. Regardless of the hardware specifications, the cost of a Diffie-Hellman key agreement (1024 bits) is 2160000 cycles. On the other hand, the Diffie-Hellman key agreement (1024 bits) can be considered equivalent to the general exponentiation in G_3 on a supersingular curve. Table 2, shows the cost of different operations.

Table 2. Cost of different operations based on cycle (with 2 decimal places)

Operation	Symbol	Cost (cycle)
Pairing	P	21600000
General exponentiation / general scalar multiplication	EXP/SM	2160000
Hashing into G_1 or G_2	H	2160000
Fixed-base exponentiation / fixed-point scalar multiplication	EXP_f/SM_f	432000
Point addition	PA	9947.37
Modular multiplication	MUL	710.53
Encryption / decryption in Z_p	E_{160}/D_{160}	512
Encryption / decryption in G_1 or G_2	E_{512}/D_{512}	1024
Encryption / decryption in G_3	E_{1024}/D_{1024}	2048

As mentioned in Section 3, Chen *et al.* in [11] first proposed a scheme to outsource pairing operations. Then, using their scheme and the scheme presented in [12], they outsourced the BF encryption and decryption schemes. In [11] and [12], communication channels between the client and the servers are considered secure. Therefore, for an accurate comparison, the cost of securing communication channels should be added to the computational complexity of the schemes presented in [11] and [12]. In addition, the client in the scheme presented in [12], must perform two inverse operations, which can be skipped by making minor changes in the pre-computation phase.

Our proposed schemes in the pre-computation phase are significantly more efficient than those presented in [11]. We have already explained that to eliminate inverse operations from the online computational complexity of the client in the schemes presented in [11], inversion must be done in the pre-computation phase. Accordingly, in Table 3, we have compared the computational complexity of the pre-computational phase of our schemes with similar schemes presented in [11].

Table 4 compares the $BFEnc\_O$ and $BFDec\_O$ schemes with the standard BF encryption and decryption schemes (without outsourcing) as well as the similar schemes presented in [11] in terms of client and server computational complexity and checkability. Without outsourcing, the checkability does not make any sense, since all the calculations are done locally by the client. The checkability of the $BFEnc\_O$ scheme depends on the parameter σ and is adjustable. If we consider $\sigma = 1$ to be checkability equal to the similar scheme presented in [11], the better efficiency of our scheme is visible. Also, the $BFDec\_O$ scheme, despite having checkability 1, is more efficient than the similar scheme presented in [11].

Table 3. Comparison of the pre-computation phase of our proposed schemes for outsourcing BF encryption and decryption schemes with similar schemes presented in [11] (the inverse operation is indicated by the symbol INV)

BF cryptography		Pre-computation phase complexity
Encryption	Ref [11]	$3P + 19SM + 10INV$
	$BFEnc\_O$ scheme	$1P + 1SM + (2\sigma + 1)E_{160}$
	$BFEnc\_O$ scheme ($\sigma = 1$)	$1P + 1SM + 3E_{160}$
Decryption	Ref [11]	$3P + 14SM + 5INV$
	$BFDec\_O$ scheme	$1EXP + 2SM + 2INV + 1E_{160}$

A comparison of the computational complexity of $BLMQSign\_O$ and $BLMQVerify\_O$ schemes with the standard BLMQ signature generation and signature verification schemes is performed in Table 5. In the $BLMQVerify\_O$ scheme, the client can use $-h$ instead of h during outsourcing $e(Q_1, Q_2)^h$, without needing to compute the inverse of h. It should be noted that in the $BLMQSign\_O$ scheme, during outsourcing $(r + h)K_{ID}$, the server must perform a general scalar multiplication because the private key is masked. Nevertheless, if the client wants to compute $(r + h)K_{ID}$ locally, it must do a fixed-point scalar multiplication.

6.4 Communication Complexity

In this section, we calculate the client communication complexity in our proposed schemes and compare them with similar schemes. It should be noted that the criterion for client communication complexity is the number of transmission bits between the client and the servers. We know the elements of Z_p are 160-bits. In addition, to transmit any point of G_1 or G_2, we must send 1024 bits, because 512 bits are needed to transmit the x coordinate and another 512 bits to transmit the y coordinate. Note that, in practice, any point can be transmitted just by 513 bits: 512 bits for x coordinate and one bit to determine whether y is positive or negative. Then the server can easily compute the y coordinate using these 513 received bits. Also, elements of G_3 are 1024-bits. Moreover, elements of Z_p encrypted using the AES are extended from 160 bits to 256 bits (two 128-bit blocks). According to the above description, we have computed the client communication complexity in our proposed schemes in Table 6 and Table 7.

7 Conclusion

In this study, we present four schemes for secure outsourcing of BF IBE and BLMQ signature schemes in the OMTUP model. The checkability of the BF decryption and BLMQ signature generation and verification schemes are 1. However, the checkability of the BF encryption scheme is based on the parameter σ and can be tuned by the client. In fact, for greater checkability, the larger σ should be selected, but it

Table 4. Comparison of client and server computational complexity and checkability of our proposed schemes for outsourcing BF encryption and decryption schemes with similar schemes presented in [11] and standard BF encryption and decryption schemes

BF cryptography		**Client online computational complexity**		**The computational complexity of each server**		**Checkability**
		Symbol	**Cycle**	**Symbol**	**Cycle**	
Encryption	Without outsourcing	$2SM_f + 1P$	22464000	-	-	-
	Ref [11]	$13PA + 10MUL + 12E_{160} + 16E_{512} + 12D_{512} + 8D_{1024}$	187621.11	$6SM + 4P + 6D_{160} + 6E_{512} + 8D_{512} + 4E_{1024}$	99385600	$\frac{1}{2}$
	BFEnc_O scheme	$3PA + 1E_{160} + (2 + 2\sigma)D_{512} + 1MUL$	31064.64 + $(\sigma + 1)$2048	$(\sigma + 2)SM_f + (\sigma + 1)(D_{160} + E_{512}) + 1P$	22032000 + $(\sigma + 1)$433536	$\frac{\sigma}{\sigma+1}$
	BFEnc_O scheme ($\sigma = 1$)	$3PA + 1E_{160} + 4D_{512} + 1MUL$	35160.64	$3SM_f + 2D_{160} + 2E_{512} + 1P$	22899072	$\frac{1}{2}$
Decryption	Without outsourcing	$1SM_f + 1P$	22032000	-	-	-
	Ref [11]	$9PA + 7MUL + 6E_{160} + 16E_{512} + 6D_{512} + 8D_{1024}$	136484.04	$3SM + 4P + 3D_{160} + 3E_{512} + 8D_{512} + 4E_{1024}$	92900992	$\frac{1}{2}$
	BFDec_O scheme	$1PA + 3MUL + 1E_{160}$	12590.96	$1SM_f + 1EXP + 1P + 1D_{160}$	24192512	1

Table 5. Comparison of client and server computational complexity and checkability of our proposed schemes for outsourcing BLMQ signature generation and verification schemes with standard BLMQ signature generation and verification schemes

BLMQ signature		**Client online computational complexity**		**The computational complexity of each server**		**Checkability**
		Symbol	**Cycle**	**Symbol**	**Cycle**	
Signature generation	Without outsourcing	$1EXP_f + 1SM_f$	864000	-	-	-
	BLMQSign_O scheme	$1MUL$	710.53	$1SM$	2160000	1
Signature verification	Without outsourcing	$1SM_f + 1PA + 1P + 1EXP_f + 1MUL$	22474657.9	-	-	-
	BLMQVerify_O scheme	$1PA + 1MUL$	10657.9	$1SM_f + 1P + 1EXP_f$	22464000	1

Table 6. Comparison of the client communication complexity of our proposed schemes for outsourcing BF encryption and decryption schemes with similar schemes presented in [11]

BF cryptography		**Client communication complexity (bit)**
Encryption	Ref [11]	52224
	BFEnc_O scheme[a]	15168 + (2560)σ
	BFEnc_O scheme ($\sigma = 1$)	17728
Decryption	Ref [11]	38400
	BFDec_O scheme	15168

[a] In step 5 of the *BFEnc_O* scheme, M can be sent to the servers only once instead of $\sigma + 1$.

Table 7. Client communication complexity of outsourcing BLMQ signature generation and verification schemes

BLMQ signature	**Client communication complexity (bit)**
BLMQSign_O scheme	4416
BLMQVerify_O scheme	14976

should be noted that the computational complexity also increases. We outsourced all schemes seamlessly. In other words, for outsourcing the pairing, scalar multiplication, and modular exponentiation operations in each scheme, we did not use the previously presented schemes for outsourcing the mentioned operations as a subroutine. As a result, due to integrated outsourcing tailored to the features of each operation, the computational and communication complexity of our schemes for outsourcing BF encryption and decryption schemes is less than similar existing schemes. In addition, contrary to previous schemes, we considered communication channels insecure and considered the cost of securing communications in the computational complexity of the client and servers. We also outsourced the BLMQ signature scheme for the first time in this paper. In future works, we can outsource other standard identity-based schemes, such as the Boneh-Boyen encryption scheme. We also aim to propose secure schemes for outsourcing the BF and the BLMQ primitives in the OUP model where the servers do not collude with each other.

References

[1] Mohammad Reza Saeidi and Hamid Mala. Secure Outsourcing of Boneh-Franklin Identity-Based Encryption Scheme. In *2021 18th International ISC Conference on Information Security and Cryptology (ISCISC)*, pages 42–49. IEEE, 2021.

[2] Haibo Tian, Fangguo Zhang, and Kui Ren. Secure bilinear pairing outsourcing made more efficient and flexible. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 417–426, 2015.

[3] Dustin Moody, Rene Peralta, Ray Perlner, Andrew Regenscheid, Allen Roginsky, and Lily Chen. Report on pairing-based cryptography. *Journal of research of the National Institute of Standards and Technology*, 120:11, 2015.

[4] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[5] IEEE Standard for Identity-Based Cryptographic Techniques using Pairings. *IEEE Std 1363.3-2013*, pages 1–151, 2013.

[6] Patrick P Tsang, Sherman SM Chow, and Sean W Smith. Batch pairing delegation. In *International Workshop on Security*, pages 74–90. Springer, 2007.

[7] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press,

2005.

[8] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[9] Benoit Chevallier-Mames, Jean-Sébastien Coron, Noel McCullagh, David Naccache, and Michael Scott. Secure delegation of elliptic-curve pairing. In *International Conference on Smart Card Research and Advanced Applications*, pages 24–35. Springer, 2010.

[10] Bo Gyeong Kang, Moon Sung Lee, and Je Hong Park. Efficient delegation of pairing computation. *Cryptology ePrint Archive*, 2005.

[11] Xiaofeng Chen, Willy Susilo, Jin Li, Duncan S Wong, Jianfeng Ma, Shaohua Tang, and Qiang Tang. Efficient algorithms for secure outsourcing of bilinear pairings. *Theoretical Computer Science*, 562:112–121, 2015.

[12] Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, and Wenjing Lou. New algorithms for secure outsourcing of modular exponentiations. *IEEE Transactions on Parallel and Distributed Systems*, 25(9):2386–2396, 2013.

[13] Li-Hua Liu and Zheng-Jun Cao. A note on efficient algorithms for secure outsourcing of bilinear pairings. *International Journal of Electronics and Information Engineering*, 5(1):30–36, 2016.

[14] Öznur Arabacı, Mehmet Sabir Kiraz, Osmanbey Uzunkol, et al. More efficient secure outsourcing methods for bilinear maps. *Cryptology ePrint Archive*, 2015.

[15] Yuchuan Luo, Shaojing Fu, Kai Huang, Dongsheng Wang, and Ming Xu. Securely outsourcing of bilinear pairings with untrusted servers for cloud storage. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 623–629. IEEE, 2016.

[16] Xi-Jun Lin, Haipeng Qu, and Xiaoshuai Zhang. New efficient and flexible algorithms for secure outsourcing of bilinear pairings. *Cryptology ePrint Archive*, 2016.

[17] Yanli Ren, Ning Ding, Tianyin Wang, Haining Lu, and Dawu Gu. New algorithms for verifiable outsourcing of bilinear pairings. *Science China Information Sciences*, 59(9):1–3, 2016.

[18] Le Tong, Jia Yu, and Hanlin Zhang. Secure outsourcing algorithm for bilinear pairings without pre-computation. In *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–7. IEEE, 2019.

[19] Hanlin Zhang, Le Tong, Jia Yu, and Jie Lin. Blockchain-Aided Privacy-Preserving Outsourcing Algorithms of Bilinear Pairings for Internet of Things Devices. *IEEE Internet of Things Journal*, 8(20):15596–15607, 2021.

[20] Jiaxiang Yang, Yanping Li, and Yanli Ren. Novel and Secure Outsourcing Algorithms for Multiple Bilinear Pairings with Single Untrusted Server. *Int. J. Netw. Secur.*, 21(5):872–880, 2019.

[21] Chih-Hung Wang and Guo-Cyuan Mao. Secure and flexible algorithm for outsourcing of bilinear pairings effectively resisting conspiracy. In *2020 15th Asia Joint Conference on Information Security (AsiaJCIS)*, pages 40–45. IEEE, 2020.

[22] Chao Lin, Debiao He, Xinyi Huang, Xiang Xie, and Kim-Kwang Raymond Choo. Blockchain-based system for secure outsourcing of bilinear pairings. *Information Sciences*, 527:590–601, 2020.

[23] Kai Zhou and Jian Ren. Secure outsourcing of scalar multiplication on elliptic curves. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2016.

[24] Kai Zhou, MH Afifi, and Jian Ren. ExpSOS: secure and verifiable outsourcing of exponentiation operations for mobile cloud computing. *IEEE Transactions on Information Forensics and Security*, 12(11):2518–2531, 2017.

[25] Yuan Ping, Xuyang Guo, Baocang Wang, and Jingxian Zhou. Secure outsourcing of modular inverses and scalar multiplications on elliptic curves. *International Journal of Security and Networks*, 15(2):101–110, 2020.

[26] Zhequn Zhao, Hanlin Zhang, Chi Zhang, Jie Lin, Jin Guo, and Yalong Wu. Privacy-preserving Outsourcing Algorithm for Finding the Shortest Path of Non-negative Weight Graphs. *International Core Journal of Engineering*, 8(3):562–576, 2022.

[27] Hongjun Li, Fanyu Kong, Jia Yu, Hanlin Zhang, and Yunting Tao. Privacy-Preserving and Verifiable Outsourcing Message Transmission and Authentication Protocol in Iot with Edge Computing. *Available at SSRN 4080721*.

[28] Wenjing Gao, Jia Yu, Ming Yang, and Huaqun Wang. Enabling Privacy-Preserving Parallel Outsourcing Matrix Inversion in IoT. *IEEE Internet of Things Journal*, 2022.

[29] Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In *Theory of cryptography conference*, pages 264–282. Springer, 2005.

[30] Xavier Boyen. A tapestry of identity-based encryption: practical frameworks compared. *International Journal of Applied Cryptography*, 1(1):3–21, 2008.

[31] Wei Dai. Crypto++ 5.6.0 Benchmarks. `https://www.cryptopp.com/benchmarks.html`, Mar 2009. Accessed: 2021-06-15.

[32] Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.

Mohammad Reza Saeidi received his B.Sc. degree in information technology (IT) engineering from Isfahan University of Technology (IUT) in 2016, and his M.Sc. degree in IT engineering (information security) from the University of Isfahan (UI) in 2018. He is currently a Ph.D. student in IT engineering (information security) at UI. His research interests include security protocols, secure outsourcing, and post-quantum cryptography.

Hamid Mala received his B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from Isfahan University of Technology (IUT) in 2003, 2006, and 2011, respectively. He joined the Department of Information Technology Engineering, University of Isfahan (UI) in September 2011, as an assistant professor. He is currently with the Faculty of Computer Engineering, UI, as an associate professor. His research interests include the design and cryptanalysis of block ciphers, cryptographic protocols, and secure multiparty computation.