mathematics

MDPI

Article

Kyber AHE: An Easy-to-Implement Additive Homomorphic Encryption Scheme Based on Kyber and Its Application in Biometric Template Protection

Roberto Román *, Rosario Arjona and Iluminada Baturone

Instituto de Microelectrónica de Sevilla (IMSE-CNM), Universidad de Sevilla, CSIC, 41092 Seville, Spain; marjona@us.es (R.A.); lumi@us.es (I.B.)
* Correspondence: rrhajderek@us.es

Abstract

Homomorphic encryption solutions tend to be costly in terms of memory and computational resources, making them difficult to implement. In this paper, we present Kyber AHE, a lightweight additive homomorphic encryption scheme for computing the addition modulo 2 of two binary strings in the encrypted domain. It is based on the CRYSTALS-Kyber public key encryption (PKE) scheme, which is the basis of the NIST module-lattice-based key-encapsulation mechanism standard. Apart from being quantum-safe, Kyber PKE has other interesting features such as the use of compressed ciphertexts, reduced sizes of keys, low execution times, and the ability to easily increase the security level. The operations performed in the encrypted domain by Kyber AHE are the decompression of ciphertexts, the component-wise modulo q addition of polynomials, and the compression of the results. A great advantage of Kyber AHE is that it can be easily implemented along with CRYSTALS-Kyber without the need for additional libraries. Among the applications of homomorphic encryption, biometric template protection schemes are a promising solution to provide data privacy by comparing biometric features in the encrypted domain. Therefore, we present the application of Kyber AHE for the protection of biometric templates. Experimental results have been obtained using Kyber AHE in an iris biometric template protection scheme with 256-byte features using Kyber512, Kyber768, and Kyber1024 instances. The sizes of the encrypted iris features are 6.0, 8.5, and 12.5 kB for NIST security levels I, III, and V, respectively. Using a commercial laptop, the encryption ranges from 0.755 to 1.73 ms, the evaluation from 0.096 to 0.161 ms, and the decryption from 0.259 to 0.415 ms, depending on the security level.

Keywords: post-quantum cryptography; CRYSTALS-Kyber; homomorphic encryption; biometric template protection

MSC: 68P25

check for updates

Academic Editors: Zhaoquan Gu and Jianxin Li

Received: 17 July 2025
Revised: 26 August 2025
Accepted: 2 September 2025
Published: 9 September 2025

Citation: Román, R.; Arjona, R.; Baturone, I. Kyber AHE: An Easy-to-Implement Additive Homomorphic Encryption Scheme Based on Kyber and Its Application in Biometric Template Protection. *Mathematics* **2025**, *13*, 2914. https://doi.org/10.3390/math13182914

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Homomorphic encryption schemes are a promising tool to perform computations in a private way without revealing sensitive information. However, homomorphic encryption schemes are not generally easy to implement, which limits their widespread use. Also, the more complex a scheme is to implement, the more difficult it is to protect against side-channel attacks because it is more complicated to control the security of the multiple

components. For example, side-channel vulnerabilities have been found in Microsoft's SEAL library [1–4].

When we talk about ease of implementation, we also include the choice of parameters, and this has a relation with the definition of security strength. It seems that the uncertainty about the discovery of new quantum algorithms and the limited ability to predict the performance of future quantum computers makes it difficult to define the strength of a given cryptosystem. This is mentioned in Section 4 of the Call for proposals of the NIST post-quantum competition [5]. This makes the cryptographic engineering of quantum-safe cryptographic primitives a messy task. A curious example is that most quantum-safe homomorphic encryption schemes express the security of the scheme in bits, while the call for proposals of the NIST post-quantum competition does not agree with the definition of the security in bits. Instead, they state that the security should be expressed using a category (I to V) defined by a comparatively easy-to-analyze reference symmetric primitive whose security serves as a floor. Thus, there is a lack of homogeneity in defining the security of quantum-safe primitives.

Biometrics is a promising technology that can be used in conjunction with other authentication methods, such as passwords or tokens, to build strong multi-factor authentication systems for identity management. However, by their very nature (immutable and unique to each individual), biometric traits are sensitive information. This is reflected in the European (EU) General Data Protection Regulation 2016/679 [6]. For this reason, biometric data should be processed with additional schemes to ensure their protection. These schemes are called biometric template protection schemes.

One approach that has gained attention for the protection of biometric templates is the use of homomorphic encryption schemes. In an enrollment phase, a biometric reference (template) is stored in encrypted form. Then, in an authentication phase, a fresh biometric query is generated by the user and later encrypted and compared with the reference in the encrypted domain. The result of the comparison in the encrypted domain is the same as the result of the comparison in the unencrypted domain. Several approaches have been proposed to achieve this [7–9].

Homomorphic encryption schemes that allow an unlimited number of operations on an unlimited number of ciphertexts are called Fully Homomorphic Encryption schemes [10]. If there is a limit on the number of operations, the scheme is called a Somewhat Homomorphic Encryption scheme [11]. These types of schemes have been tested in the literature for biometric template protection. Examples include BFV and CKKS, which work with floating-point and integer numbers, respectively.

On the other hand, there is an interest in the literature for homomorphic encryption systems that work with binary numbers, since they generally achieve better performance. Biometric features can be binarized without significantly reducing recognition accuracy [12–14]. Also, biometric features can be obtained directly in binary form, such as iris, veins, or palm print features. Note that the privacy of iris features has raised concerns in the public domain, and new works are attempting to address this [15].

1.1. Motivations

We have seen that homomorphic encryption schemes are proposed to be universal solutions that are not tied to concrete applications. However, one drawback of homomorphic encryption schemes is its low performance and even impracticability in certain situations. For example, the work in [16] directly used the CKKS scheme in a fingerprint biometric template protection, and the authors concluded that this scheme was not practical.

We think that the proposal of new homomorphic encryption schemes tied to specific settings is an interesting line of research. In this work, we focus on dealing with messages

that have a binary format, and we are interested in the setting where an addition modulo 2 between two binary strings is computed in the encrypted domain and not directly as plaintext. The addition modulo 2 is the same as the XOR operation and can be used to compute the Hamming distance without affecting biometric accuracy. Biometric systems are a clear application where high performance is required because the number of users can be significant.

The work in [17] uses the NTRUEncrypt scheme presented in [18] as an additive homomorphic encryption scheme to protect binary biometric templates. However, the version of NTRUEncrypt used is not provably secure [19]. This renders NTRUEncrypt undesirable as a homomorphic encryption scheme, since at least IND-CPA security is required. Even provably secure versions of the scheme, such as the one in [19], have been shown to have certain vulnerabilities [20,21]. Recently used, NTRU-based public key encryption schemes do not seem to be homomorphic [22]. At the time of writing, it is unclear whether the NTRU cryptosystem could be used to develop a provably secure additive homomorphic encryption scheme. Remarkably, none NTRU-based proposal was selected in the NIST post-quantum competition [23].

1.2. Contributions

In this paper, we propose a homomorphic encryption scheme based on CRYSTALS-Kyber Public Key Encryption, called Kyber Additive Homomorphic Encryption (Kyber AHE). Why Kyber? The reason is that it is a scheme already used in the CRYSTALS-Kyber Key Encapsulation Mechanism (KEM), which is standardized in FIPS-203 [24,25]. It is expected that Kyber KEM will be widely used to exchange ephemeral keys and achieve secure communication. For example, it is expected that it will be used in quantum-safe TLS 1.3 [26,27]. The Call for proposals of the NIST post-quantum competition defines a well-established framework for evaluating the strength of the security of candidates. Thus, Kyber comes with a well-defined notion of security that makes the parameter selection of Kyber AHE easy and homogeneous with existing cryptographic primitives (as the recently standardized Kyber KEM).

Kyber has also some interesting features such as the use of compressed ciphertexts, low execution times, and the ability to easily increase the security level. There are some efforts in the literature to use Kyber for face template protection. A preliminary work was presented in [28]. However, in [28] the homomorphic encryption scheme was not defined, the concrete operations in the encrypted domain were not explained in detail, the homomorphic property of Kyber was not formally proved, and an analysis of the selection of concrete parameters was not shown. We plan to fill this gap with this paper.

Note that the proposed scheme is not general, and we cannot add an arbitrary number of values or perform other binary operations, such as AND. However, the scheme is very lightweight for its intended purpose. In addition, it can be easily integrated with Kyber KEM, for example, in a Trusted Execution Environment (TEE) with constrained enclave sizes [29].

The specific contributions are the following:

1. We present a concrete definition of Kyber AHE, with algorithms for key generation, encryption, evaluation, and decryption. The IND-CPA security of Kyber AHE, required in homomorphic encryption schemes, is reduced to the IND-CPA security of Kyber PKE.
2. We demonstrate the homomorphic property of the scheme with a theorem, which is used to estimate the failure probability of the scheme in computing the addition modulo 2 of two binary strings. This is important because Kyber AHE (as well as Kyber PKE) algorithms are probabilistic. The results show that very low failure

probabilities can be achieved using the parameters of the standardized instances Kyber512, Kyber768, and Kyber1024.

3. We are the first in the literature to propose a homomorphic encryption scheme with parameters related to security levels according to the NIST call for proposals. This is because we can use the standardized Kyber instance.
4. We show how to apply the proposed scheme in a biometric template protection setting and present concrete experimental results using an iris biometric feature as a reference, specifically the well-known IrisCodes. The performance parameters evaluated are the size of the keys (public and private), the size of the encrypted iris feature, and the times taken to encrypt, evaluate an addition modulo 2 between two binary strings, and decrypt the result.

1.3. Organization

The paper is organized as follows. First, the biometric template protection setting and Kyber PKE are explained as preliminaries in Section 2. Then, the additive homomorphic encryption scheme Kyber AHE is presented in Section 3, where the homomorphic property is demonstrated, and a failure probability analysis is performed (Contributions 1 and 2). The security of the scheme is also discussed, and suitable parameter sets are identified by analyzing the failure probability (Contribution 3). Section 4 shows the performance results of the proposal for the protection of iris features (Contribution 4). Finally, Section 5 concludes the paper.

2. Preliminaries

2.1. Biometric Template Protection Setting with Homomorphic Encryption

We consider the biometric template protection of features in binary form that are compared using the Hamming distance, i.e., addition modulo 2. An example is the iris features. We use the scheme in [30,31] as references of homomorphic encryption-based settings, although they employ non-standardized post-quantum cryptography (among others, CKKS and BFV in the case of [30], and ideal lattices in the case of [31]).

We assume three parties, the Client, the Database Server, and the Matching Server. It is assumed a *passive honest-but-curious* model where the servers will not deviate from the defined protocol but will attempt to learn all possible information from legitimately received messages. Also, the servers cannot collude, that is, exchange information outside the protocol. This is often assumed in biometric template protection schemes based on homomorphic encryption [7,30]. If the non-collusion assumption is not met, the database server can send the user's encrypted biometric feature to the matching server. This allows them to know the biometric feature without deviating from the protocol. This could affect the user's security and privacy in two ways. First, if an attacker gains access to one of the servers and learns the user's biometric features, s/he can impersonate the user in other applications. This is a major concern since biometric features are linked to individuals and cannot be changed. Second, if the database server knows the biometric feature in clear text, the user can be traced if different enrollments have been performed in more than one matching server. Thus, the user privacy is affected. Of course, if they collaborate, the matching servers could also trace the user.

It is assumed that the communication between the entities is secure. KEMTLS, a quantum-safe upgrade of TLS 1.3, and Kyber KEM can be used to establish secure and authenticated channels [26]. First, in an enrollment phase, a reference biometric template is stored in an encrypted way with Kyber AHE (with the algorithm denoted as Kyber.AHE.Encrypt) in the Database Server. In an authentication phase, shown in Figure 1, the user sends a fresh biometric query encrypted with Kyber AHE to the Database Server,

and the Database Server then runs the Kyber AHE evaluation algorithm (denoted as Kyber.AHE.Eval) on the enrolled ciphertext and the fresh ciphertext. Finally, the resulting ciphertext is sent to the Matching Server and decrypted (with the algorithm denoted as Kyber.AHE.Decrypt) to obtain the Hamming distance between the biometric features. The algorithms of Kyber AHE are detailed in Section 3.

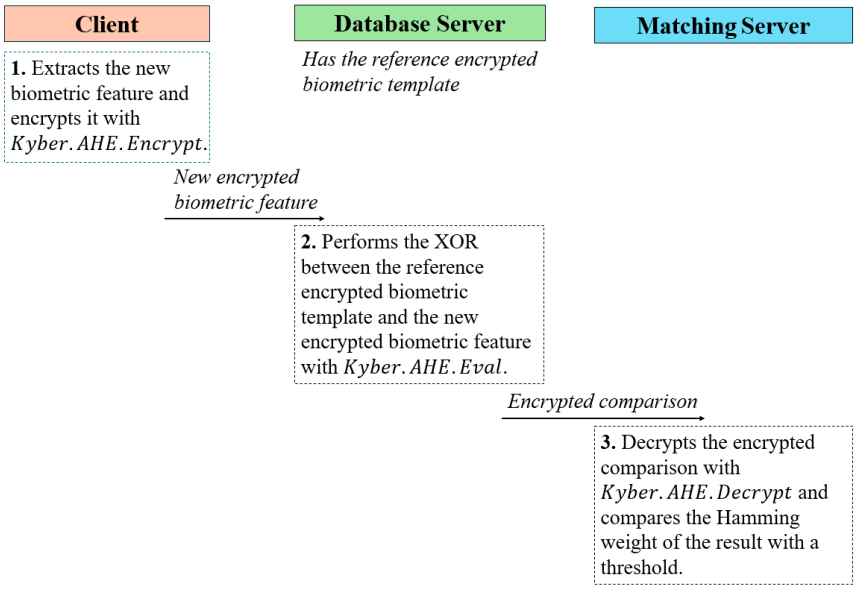


Figure 1. Use of Kyber AHE in the authentication phase of a biometric template protection scheme.

Note that, with this scheme, the matching server knows more than just the Hamming distance because it knows the XOR between two biometric responses. This is an issue already reported in the literature [32]. It has been stated that the XOR of an individual's biometric responses can reveal information about the reliability of those responses and that this information could be used to track the individual. However, several solutions have been proposed that could use the proposed scheme while circumventing this issue. One solution is to use Biohashing and cancellable biometrics to erase the link between the template and the individual [32]. Promising results have been found for face templates [33]. Another solution is the use of multi-party computation (MPC) and garbled circuits [34].

2.2. Applications of Additive Homomorphic Encryption Beyond Biometric Template Protection

Masking against side-channel attacks. Side-channel attacks are a major concern for lattice-based cryptographic schemes, including Kyber [35]. To address this issue, new masking techniques are being developed to protect specific implementations. One interesting masking scheme for ring-LWE decryption is presented in [36]. The main idea is that an additive homomorphism in the encryption scheme can introduce randomness into the decryption procedure. Suppose we want to decrypt the ciphertext $c = Enc(m, sk)$, where m is the message that we want to decrypt. First, an internal random message m' is generated. Then, m' is encrypted using the secret key sk to obtain $c' = Enc(m', sk)$. Later, the resulting ciphertext c' is homomorphically added to c, resulting in $c' + c$. Finally, $c' + c$ is decrypted to obtain $m \oplus m'$, which, XOR-ed with m', is used to obtain the message m. The mentioned work applies the idea to the LPR (Lyubashevsky, Peikert, and Regev) encryption scheme.

The aforementioned approach is easy to implement because it does not require a masked decoder. However, the experimental results show that the failure rate is too high, making the solution impractical. For example, it cannot be used with CCA-secure schemes [37]. In this paper, we demonstrate that the parameters used with Kyber AHE pro-

vide low failure probabilities. Thus, this work opens the door to new masking techniques for Kyber PKE or Kyber KEM.

Other schemes that use non-uniform and unreliable secrets. Additive homomorphic encryption can be applied in scenarios where a string that is neither completely uniform nor reliable is used to identify an entity. Device authentication using Physically Unclonable Functions (PUFs) is a good example. PUFs measure the effects of random process variations during the manufacturing of integrated circuits. These variations result in a binary string that can be used as an identifier. Thus, PUF data is similar to biometric data, and the same approach used to compute the XOR of a reference and a fresh response in the encrypted domain can be applied [38,39].

2.3. Notation

Rings and vectors. Let $\mathbb{Z}$ denote the ring of integers and $\mathbb{Z}_q$ denote the ring of integers modulo an integer q. R and R_q denote the rings $\frac{\mathbb{Z}[X]}{X^n+1}$ and $\frac{\mathbb{Z}_q[X]}{X^n+1}$, respectively, where $n = 2^{n'-1}$ such that $X^n + 1$ is the $2^{n'}$-th cyclotomic polynomial. Throughout this paper, the values of n, n', and q are 256, 9, and 3329, respectively, as considered in the specifications of CRYSTALS-Kyber. The notation employed is that regular font letters denote elements in R or R_q (which includes elements in $\mathbb{Z}$ and $\mathbb{Z}_q$), bold lower-case letters represent vectors with components in R or R_q, and bold upper-case letters are matrices. All vectors are column vectors. For a vector $\boldsymbol{t}$ (or matrix $\boldsymbol{A}$), we denote its transpose by $\boldsymbol{t^T}$ (or $\boldsymbol{A^T}$).

Modulo reduction. For an even positive integer α, we define $r' = r \ mod^{\pm} \ \alpha$ to be the unique element r' in the range $-\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$ such that $r' = r \ mod \ \alpha$. For an odd positive integer α, we define $r' = r \ mod^{\pm} \ \alpha$ to be the unique element r' in the range $-\frac{\alpha-1}{2} < r' \leq \frac{\alpha-1}{2}$ such that $r' = r \ mod \ \alpha$. For any positive integer α, we define $r' = r \ mod^{+} \alpha$ to be the unique element r' in the range $0 \leq r' < \alpha$ such that $r' = r \ mod \ \alpha$. If the exact representation is not important, we will use the expression $r \ mod \ \alpha$.

Rounding. For an element $x \in \mathbb{Q}$ we denote $\lceil x \rfloor$ by the rounding of x to the closest integer with ties being rounded up. As example that will appear in this paper, if q is a prime number greater than two, then $\lceil \frac{q}{2} \rfloor = \frac{q+1}{2}$.

Size of elements and norms. For an element $w \in \mathbb{Z}_q$, we write $\|w\|_\infty$ to mean $\left|w \ mod^{\pm} q\right|$. For a polynomial $w \in R$ of degree n, we define the l_∞ norm for $w = \sum_{i=0}^{n-1} w_i \cdot X^i \in R$ as $\|w\|_\infty = max_{i \epsilon \{0,\ldots,n-1\}} \|w_i\|_\infty$. Similarly, for a vector of polynomials $\boldsymbol{w} = (w_1, \ldots, w_k) \in R^k$, we define it as $\|\boldsymbol{w}\|_\infty = max_{i \epsilon \{1,\ldots,k\}} \|w_i\|_\infty$.

Sets and distributions. For a set S, we write $s \leftarrow S$ to denote that s is chosen uniformly at random from S. If S is a probability distribution, then this denotes that s is chosen according to the distribution S.

Let Sam be an extendable output function (XOF), that is, a function on bit strings where the output can be extended to any length. If we want Sam to take as input x and then produce a value y that is distributed according to a given distribution S (or uniformly over a set S), we write $y \sim S \coloneqq Sam(x)$.

We define the output of a centered binomial distribution B_η for some positive integer η as $\sum_{i=1}^{\eta}(a_i - b_i)$ given the samples $\{(a_i, b_i)\}_{i=1}^{\eta} \leftarrow \left(\{0,1\}^2\right)^{\eta}$.

If v is an element of R, we write $v \leftarrow \beta_\eta$ to mean that $v \in R$ is generated from a distribution where each of its coefficients is generated according to B_η. Similarly, a k-dimensional vector of polynomials $\boldsymbol{v} \in R^k$ can be generated according to the distribution β_η^k.

Compression and decompression functions. For an element $x \in \mathbb{Z}_q$ and two positive integers d and q, with $2^d < q$, the functions $Compress_q$ and $Decompress_q$ are defined as $Compress_q(x,d) = \left\lceil \left(\frac{2^d}{q}\right) \cdot x \right\rfloor mod^{+} \ 2^d$ and $Decompress_q(x,d) = \left\lceil \left(\frac{q}{2^d}\right) \cdot x \right\rfloor$. An important

property (demonstrated in Appendix X.C in [40]) is that if $x' = Decompress_q\Big(Compress_q(x,d),d\Big)$, then the following is met:

$$\left|x' - x \; mod^{\pm} \; q\right| = \left|x \; - x' mod^{\pm} \; q\right| \leq \left\lceil \frac{q}{2^{d+1}} \right\rfloor \tag{1}$$

When $Compress_q$ or $Decompress_q$ is used with $x \in R_q$ (and $\boldsymbol{x} \in R_q^k$), the procedure is applied to each coefficient individually (and each polynomial individually).

2.4. The Module-LWE Problem

The security of Kyber PKE is based on the hardness of the Module Learning With Errors (Module-LWE) problem [24]. The decision Module-LWE problem is defined as follows.

Definition 1 (Decision Module-LWE)**.** *Let k and q be integer parameters. The problem states that it is hard to distinguish samples* $(\boldsymbol{a}_i, b_i) \in R_q^k \times R_q$ *where the coefficients of* $\boldsymbol{a}_i$ *and* b_i *are sampled from a uniform random distribution, from samples* $(\boldsymbol{a}_i, b_i) \in R_q^k \times R_q$*, where the coefficients of* $\boldsymbol{a}_i$ *are sampled from a uniform random distribution and* $b_i = \boldsymbol{a}_i^T \boldsymbol{s} + e_i$ *such that* $\boldsymbol{s} \leftarrow \beta_\eta^k$ *and* $e_i \leftarrow \beta_\eta$*, where* $\boldsymbol{s}$ *is common to all samples and* e_i *is fresh.*

For an algorithm A, the adversary $\boldsymbol{Adv}_{m,k,\eta}^{MLWE}(\mathrm{A})$ against the decision MLWE problem is defined in [24] as follows:

$$\left| Pr\left[b' = 1 : \begin{pmatrix} \boldsymbol{A} \leftarrow R_q^{m \times k}; (\boldsymbol{s}, \boldsymbol{e}) \leftarrow \beta_\eta^k \times \beta_\eta^m; \\ \boldsymbol{b} = \boldsymbol{A}\,\boldsymbol{s} + \boldsymbol{e}; b' \leftarrow A(\boldsymbol{A}, \boldsymbol{b}) \end{pmatrix} \right] - Pr\left[b' = 1 : \begin{pmatrix} \boldsymbol{A} \leftarrow R_q^{m \times k}; \\ \boldsymbol{b} \leftarrow R_q^m; b' \leftarrow A(\boldsymbol{A}, \boldsymbol{b}) \end{pmatrix} \right] \right| \tag{2}$$

2.5. Kyber Public Key Encryption

The Kyber Public Key Encryption (PKE) scheme is defined by a message space M and a triple of probabilistic algorithms, which are key generation (*Kyber.PKE.KeyGen*), encryption (*Kyber.PKE.Encrypt*), and decryption (*Kyber.PKE.Decrypt*) algorithms. The key generation algorithm returns a pair (pk, sk) consisting of a public key and a secret key, respectively. The encryption algorithm takes a public key pk and a message $m \in M$ to produce a ciphertext c. Finally, the decryption algorithm takes a secret key sk and a ciphertext c and returns a message m.

In the *Kyber.PKE.KeyGen* algorithm, defined as Algorithm 1, two seeds, ρ and σ, are sampled from a uniform random distribution. These seeds are used to sample coefficients from a uniform random distribution for polynomials in the matrix $\boldsymbol{A}$ and from a binomial central distribution (β_η) for polynomials in $\boldsymbol{s}$ and $\boldsymbol{e}$, where η is a positive integer. A XOF is used. $\boldsymbol{t}$ is computed as $\boldsymbol{A}\,\boldsymbol{s} + \boldsymbol{e}$. Note that the result is not compressed because a problem was found in the security proof for the indistinguishability under chosen plaintext attack (IND-CPA) [41]. The seed ρ is also stored with $\boldsymbol{t}$ to form the public key pk and $\boldsymbol{s}$ is stored as the secret key sk.

Algorithm 1. *KYBER.PKE.KeyGen*()

1: $\rho, \sigma \leftarrow \{0,1\}^{256}$
2: $\boldsymbol{A} \sim R_q^{k \times k} := Sam(\rho)$
3: $(\boldsymbol{s}, \boldsymbol{e}) \sim \beta_\eta^k \times \beta_\eta^k := Sam(\sigma)$
4: $\boldsymbol{t} := \boldsymbol{A}\,\boldsymbol{s} + \boldsymbol{e}$
5: ***return*** $(pk := (\boldsymbol{t}, \rho),\ sk := \boldsymbol{s})$

The *Kyber.PKE.Encrypt* algorithm, defined as Algorithm 2, takes, as input, a message m, with message space $M = \{0,1\}^{256}$, and the public key $pk = (\boldsymbol{t}, \rho)$. Every message $m \in M$ can be viewed as a polynomial in R with coefficients in $\{0,1\}$. First, a seed r is

sampled from a uniform random distribution. Then, the matrix $\boldsymbol{A}$ is retrieved from the seed ρ as in the *Kyber.PKE.KeyGen* algorithm. The seed r is used for sampling coefficients from a binomial central distribution β_η for polynomials in $\boldsymbol{r}$ and $\boldsymbol{e}_1$ and the polynomial e_2. Then, two parts of the ciphertext are computed, from $\boldsymbol{A^T r} + \boldsymbol{e}_1$ and $\boldsymbol{t^T r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m$. The two parts are compressed with $Compress_q$, resulting in $\boldsymbol{u}$ and v. The ciphertext c is formed by $\boldsymbol{u}$ and v.

Algorithm 2. $KYBER.PKE.Encrypt(pk = (\boldsymbol{t}, \rho), m \in M)$

1: $r \leftarrow \{0,1\}^{256}$
2: $\boldsymbol{A} \sim R_q^{kxk} := Sam(\rho)$
3: $(\boldsymbol{r}, \boldsymbol{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := Sam(r)$
4: $\boldsymbol{u} := Compress_q(\boldsymbol{A^T r} + \boldsymbol{e}_1, d_u)$
5: $v := Compress_q(\boldsymbol{t^T r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v)$
6: ***return*** $c = (\boldsymbol{u}, v)$

In the *Decrypt* algorithm, defined as Algorithm 3, the ciphertext $c = (\boldsymbol{u}, v)$ and the secret key $sk = \boldsymbol{s}$ are taken as inputs. First, the parts of the ciphertext $\boldsymbol{u}$ and v are decompressed using $Decompress_q$, resulting in $\boldsymbol{u_D}$ and v_D. Then, the value $v_D - \boldsymbol{s^T u_D}$ is computed. The $Compress_q$ function is used to decrypt to a 1 if $v_D - \boldsymbol{s^T u_D}$ is closer to $\lceil \frac{q}{2} \rfloor$ than to 0 and decrypt it to a 0 otherwise.

Algorithm 3. $KYBER.PKE.Decrypt(sk = \boldsymbol{s}, c = (\boldsymbol{u}, v))$

1: $\boldsymbol{u_D} := Decompress_q(\boldsymbol{u}, d_u)$
2: $v_D := Decompress_q(v, d_v)$
3: ***return*** $m = Compress_q(v_D - \boldsymbol{s^T u_D}, 1)$

3. Kyber Additive Homomorphic Encryption

3.1. Definition

We define the Kyber Additive Homomorphic Encryption (Kyber AHE) scheme, also referred to as *Kyber.AHE*, with the message space M and the tuple of algorithms Π = (*Kyber.AHE.KeyGen*, *Kyber.AHE.Encrypt*, *Kyber.AHE.Eval*, *Kyber.AHE.Decrypt*). The algorithms are defined as follows:

- *Kyber.AHE.KeyGen*(): is the same as the *Kyber.PKE.KeyGen* algorithm defined in Section 2.5. It returns a secret key *sk* and a public key *pk*.
- *Kyber.AHE.Encrypt*(pk, m): is the same as the *Kyber.PKE.Encrypt* algorithm defined in Section 2.5. It takes the public key *pk* and the message m and returns the ciphertext $c = (\boldsymbol{u}, v)$.
- *Kyber.AHE.Eval*(c_1, c_2): it takes two ciphertexts, c_1 and c_2, and returns a ciphertext c. It works as follows. First, the two input ciphertexts, c_1 and c_2, are parsed into $(\boldsymbol{u}_1, v_1)$ and $(\boldsymbol{u}_2, v_2)$. Then, they are decompressed using the $Decompress_q$ function and added modulo q. Finally, the result is recompressed using the $Compress_q$ function. The result is the ciphertext $c = (\boldsymbol{u}, v)$. Figure 2 illustrates the high-level block diagram of the operations involved in the evaluation algorithm. The steps are shown in Algorithm 4.
- *Kyber.AHE.Decrypt*(sk, c): is the same as the *Kyber.PKE.Decrypt* algorithm defined in Section 2.5.

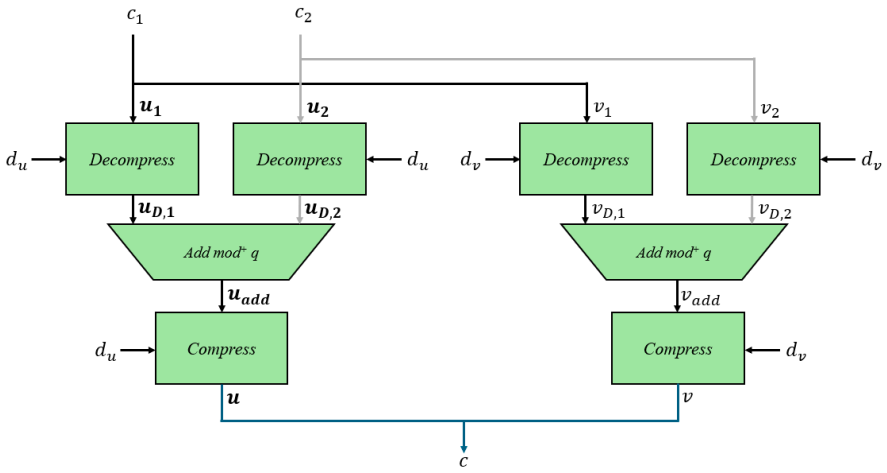


Figure 2. This illustration shows the steps performed in $Kyber.AHE.Eval(c_1, c_2)$.

This scheme allows for computing the addition modulo 2 of two messages in the encrypted domain. This is proven in Theorem 1. Note that the addition modulo 2 is the same as the coefficient-wise XOR of the messages.

Algorithm 4. $KYBER.AHE.Eval(c_1 = (\boldsymbol{u}_1,\ v_1), c_2 = (\boldsymbol{u}_2,\ v_2))$

1: $\boldsymbol{u_{D,1}} \coloneqq\ Decompress_q(\boldsymbol{u}_1, d_u)$
2: $\boldsymbol{u_{D,2}} \coloneqq\ Decompress_q(\boldsymbol{u}_2, d_u)$
3: $v_{D,1} \coloneqq\ Decompress_q(v_1, d_v)$
4: $v_{D,2} \coloneqq\ Decompress_q(v_2, d_v)$
5: $\boldsymbol{u_{add}} \coloneqq \boldsymbol{u_{D,1}} + \boldsymbol{u_{D,2}} mod^+ q$
6: $v_{add} \coloneqq v_{D,1} + v_{D,2}\ mod^+\ q$
7: $\boldsymbol{u} \coloneqq Compress_q(\boldsymbol{u}_{add}, d_u)$
8: $v \coloneqq Compress_q(v_{add}, d_v)$
9: ***return*** $(\boldsymbol{u}, v)$

Theorem 1 (Correctness). *Let* $\boldsymbol{A} \in R_q^{kxk}$, $\boldsymbol{s}$, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{e}$, $\boldsymbol{e}_{1,1}$, $\boldsymbol{e}_{1,2} \in R^k$, $e_{2,1}$, $e_{2,2} \in R$ *and let the messages* m_1, $m_2 \in R_q$ *with coefficients in* $\{0,1\}$. *It is assumed that* $q \equiv 1\ mod\ 4$. *Define:*

- $\boldsymbol{t}$, ρ, *and* $\boldsymbol{s}$ *as the output* $(\boldsymbol{t}, \rho)$ *and* $\boldsymbol{s}$ *of Algorithm 1.*
- $(\boldsymbol{u}_1, v_1)$ *as the output of Algorithm 2 applied to* m_1 *using* $\boldsymbol{t}$ *and* ρ. *Similarly,* $(\boldsymbol{u}_2, v_2)$ *as the output of Algorithm 2 applied to* m_2 *using the same* $\boldsymbol{t}$ *and* ρ.
- $(\boldsymbol{u}, v)$ *as the output of Algorithm 4 applied to* $(\boldsymbol{u}_1, v_1)$ *and* $(\boldsymbol{u}_2, v_2)$.
- $\boldsymbol{c}_{u,1}$, $\boldsymbol{c}_{u,2}$, $c_{v,1}$, *and* $c_{v,2}$ *as the compression errors introduced in Algorithm 2 to obtain* $(\boldsymbol{u}_1, v_1)$ *and* $(\boldsymbol{u}_2, v_2)$.
- $\boldsymbol{c}_u$, c_v *as the compression errors introduced in Algorithm 4 to obtain* $(\boldsymbol{u}, v)$.
- m *as the output of Algorithm 3 applied to* $(\boldsymbol{u}, v)$ *using* $\boldsymbol{s}$.

If the following is true,

$$\left\| \boldsymbol{e}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + c_{v,1} + c_{v,2} + c_v - \boldsymbol{s}^T(\boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2} + \boldsymbol{c}_u) \right\|_\infty < \left\lceil \frac{q}{4} \right\rfloor, \quad (3)$$

then, $m \in R_q$ *is equal to* $m_1 + m_2\ mod\ 2$, *that is, the XOR between the coefficients of* m_1 *and* m_2.

Proof. When Algorithm 4 is applied, the decompressed values of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are as follows:

$$\boldsymbol{u}_{D,1} = Decompress_q\Big(Compress_q\Big(\boldsymbol{A}^T\boldsymbol{r}_1 + \boldsymbol{e}_{1,1}, d_u\Big), d_u\Big) = \boldsymbol{A}^T\boldsymbol{r}_1 + \boldsymbol{e}_{1,1} + \boldsymbol{c}_{u,1} \quad (4)$$

$$\boldsymbol{u}_{D,2} = \boldsymbol{A}^T \boldsymbol{r}_2 + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,2} \tag{5}$$

where $\boldsymbol{c}_{u,1}, \boldsymbol{c}_{u,2} \in R^k$. Similarly, the decompressed values of v_1 and v_2 are as follows:

$$\begin{gathered} v_{D,1} = Decompress_q\Big(Compress_q\big(\boldsymbol{t}^T \boldsymbol{r}_1 + e_{2,1} + \lceil \tfrac{q}{2} \rfloor \cdot m_1, d_v\big), d_v\Big) = \\ \boldsymbol{t}^T \boldsymbol{r}_1 + e_{2,1} + \lceil \tfrac{q}{2} \rfloor \cdot m_1 + c_{v,1} = (\boldsymbol{A}\, \boldsymbol{s} + \boldsymbol{e})^T \boldsymbol{r}_1 + e_{2,1} + \lceil \tfrac{q}{2} \rfloor \cdot m_1 + c_{v,1} \end{gathered} \tag{6}$$

$$v_{D,2} = (\boldsymbol{A}\, \boldsymbol{s} + \boldsymbol{e})^T \boldsymbol{r}_2 + e_{2,2} + \left\lceil \frac{q}{2} \right\rfloor \cdot m_2 + c_{v,2} \tag{7}$$

In Algorithm 4, these values are added modulo q, resulting in the following:

$$\boldsymbol{u_{add}} = \boldsymbol{u_{D,1}} + \boldsymbol{u_{D,2}}\ mod^+\ q = \boldsymbol{A}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + \boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2}\ mod^+\ q \tag{8}$$

$$v_{add} = v_{D,1} + v_{D,2}\ mod^+\ q = (\boldsymbol{A}\, \boldsymbol{s} + \boldsymbol{e})^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2) + c_{v,1} + c_{v,2}\ \ mod^+\ q \tag{9}$$

Then, the values of $\boldsymbol{u}_D$ and v_D that are obtained when executing Algorithm 3 can be written as follows:

$$\boldsymbol{u}_D = Decompress_q\Big(Compress_q(\boldsymbol{u_{add}}, d_u), d_u\Big) = \boldsymbol{A}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + \boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2} + \boldsymbol{c}_u \tag{10}$$

$$v_D = Decompress_q\Big(Compress_q(v_{add}, d_v), d_v\Big) = (\boldsymbol{A}\, \boldsymbol{s} + \boldsymbol{e})^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2) + c_{v,1} + c_{v,2} + c_v \tag{11}$$

Then, the $v_D - \boldsymbol{s}^T \boldsymbol{u}_D$ term can be expressed as follows:

$$\begin{gathered} v_D - \boldsymbol{s}^T \boldsymbol{u}_D = \\ (\boldsymbol{A}\, \boldsymbol{s} + \boldsymbol{e})^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + \lceil \tfrac{q}{2} \rfloor \cdot (m_1 + m_2) + c_{v,1} + c_{v,2} + c_v \\ -\boldsymbol{s}^T\big(\boldsymbol{A}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + \boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2} + \boldsymbol{c}_u\big) = \\ \boldsymbol{e}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + \lceil \tfrac{q}{2} \rfloor \cdot (m_1 + m_2) + c_{v,1} + c_{v,2} + \\ c_v - \boldsymbol{s}^T(\boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2} + \boldsymbol{c}_u) \end{gathered} \tag{12}$$

We can conveniently accumulate all the error terms in one variable w as follows:

$$w = \boldsymbol{e}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + c_{v,1} + c_{v,2} + c_v - \boldsymbol{s}^T(\boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2} + \boldsymbol{c}_u) \tag{13}$$

Hence, Equation (12) can be expressed as follows:

$$v_D - \boldsymbol{s}^T \boldsymbol{u}_D = w + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2) \tag{14}$$

Then, the decrypted message that is obtained when Algorithm 3 is used to decrypt the result of Algorithm 4 can be expressed as follows:

$$m = Compress_q\Big(v_D - \boldsymbol{s}^T \boldsymbol{u}_D, 1\Big) \tag{15}$$

Since m can be viewed as a polynomial in R with coefficients in $\{0, 1\}$, we can write the following:

$$m' = Decompress_q(m, 1) = \left\lceil \frac{q}{2^1} \cdot m \right\rfloor = \left\lceil \frac{q}{2} \cdot m \right\rfloor = \left\lceil \frac{q}{2} \right\rfloor \cdot m = Decompress_q\Big(Compress_q(v_D - \boldsymbol{s^T u_D}, 1\Big), 1) \tag{16}$$

We also know from inequality (1) that the distance between a value and the result of the decompression of its compression has an upper bound. Thus, the following is true:

$$\left\| v_D - \boldsymbol{s}^T \boldsymbol{u}_D - m' \right\|_\infty = \left\| v_D - \boldsymbol{s}^T \boldsymbol{u}_D - \left\lceil \frac{q}{2} \right\rfloor \cdot m \right\|_\infty \leq \left\lceil \frac{q}{2^{1+1}} \right\rfloor = \left\lceil \frac{q}{4} \right\rfloor \tag{17}$$

Hence, inequality (17) can be rewritten as (using Equation (14)):

$$\left\| w + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2) - \left\lceil \frac{q}{2} \right\rfloor \cdot m \right\|_\infty = \left\| w + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) \right\|_\infty \leq \left\lceil \frac{q}{4} \right\rfloor \tag{18}$$

Then, we know from the statement of the theorem (inequality (3)) that the following is accomplished:

$$\left\| \boldsymbol{e}^{\boldsymbol{T}} (\boldsymbol{r}_1 + \boldsymbol{r}_2) + e_{2,1} + e_{2,2} + c_{v,1} + c_{v,2} + c_v - \boldsymbol{s}^{\boldsymbol{T}} (\boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{\boldsymbol{u},1} + \boldsymbol{c}_{\boldsymbol{u},2} + \boldsymbol{c}_{\boldsymbol{u}}) \right\|_\infty = \|w\|_\infty < \left\lceil \frac{q}{4} \right\rfloor \tag{19}$$

Using inequalities (18) and (19) and the triangle inequality in $\|\cdot\|_\infty$, we can also get the following inequality:

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) \right\|_\infty = \left\| w + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) - w \right\|_\infty \leq \left\| w + \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) \right\|_\infty + \|w\|_\infty < \left\lceil \frac{q}{4} \right\rfloor + \left\lceil \frac{q}{4} \right\rfloor = 2 \cdot \left\lceil \frac{q}{4} \right\rfloor \tag{20}$$

Thus, we arrive at the following inequality:

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) \right\|_\infty < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor \tag{21}$$

In order to show that the values of m, take the values of $m_1 + m_2 \ mod \ 2$, we note, first, that if m is equal to $m_1 + m_2 \ mod \ 2$; then, inequality (21) is accomplished. Then, let us imagine that the values of the coefficients of m do not take the value $m_1 + m_2 \ mod \ 2$. Then, the value of $(m_1 + m_2 - m)$ would be ± 1. In that case, there would be a coefficient with the following value:

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) \right\|_\infty = \left| \left\lceil \frac{q}{2} \right\rfloor \cdot (\pm 1) \ mod^{\pm} \ q \right| = \left| \frac{q+1}{2} \ mod^{\pm} \ q \right| = \left| \frac{-q+1}{2} \right| = \frac{q-1}{2} = 2 \cdot \frac{q-1}{4} \tag{22}$$

where $\frac{(q+1)}{2}$ is reduced to $\frac{(-q+1)}{2}$ with the $mod^{\pm}$ operation. As shown in [40], if $q \equiv 1 \ mod \ 4$, that is true since its value is 3329, then we have the equality $\left\lceil \frac{q}{4} \right\rfloor = \frac{q-1}{4}$, and the Equation (22) is the following:

$$\left\| \left\lceil \frac{q}{2} \right\rfloor \cdot (m_1 + m_2 - m) \right\|_\infty = 2 \cdot \left\lceil \frac{q}{4} \right\rfloor \tag{23}$$

which contradicts inequality (21). On the other hand, if m is equal to $m_1 + m_2 \ mod \ 2$, we have the following cases:

- $m_1 = m_2 = 0$, then $m = 0$. Inequality (21) is accomplished as follows:

$$\left| \left\lceil \frac{q}{2} \right\rfloor \cdot (0 + 0 - 0) \ mod^{\pm} \ q \right| = 0 < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor$$

- $m_1 = 0$ and $m_2 = 1$, then $m = 1$. Inequality (21) is accomplished as follows:

$$\left| \left\lceil \frac{q}{2} \right\rfloor \cdot (0 + 1 - 1) \ mod^{\pm} \ q \right| = 0 < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor$$

- $m_1 = 1$ and $m_2 = 0$, then $m = 1$. Inequality (21) is accomplished as follows:

$$\left|\left\lceil \frac{q}{2} \right\rfloor \cdot (1 + 0 - 1)\ mod^{\pm}\ q\right| = 0 < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor$$

- $m_1 = 1$ and $m_2 = 1$, then $m = 0$. Inequality (21) is accomplished as follows:

$$\left|\left\lceil \frac{q}{2} \right\rfloor \cdot (1 + 1 - 0)\ mod^{\pm}\ q\right| = \left|2 \cdot \left\lceil \frac{q}{2} \right\rfloor\ mod^{\pm}\ q\right| = |q + 1\ mod^{\pm}\ q| = 1 < 2 \cdot \left\lceil \frac{q}{4} \right\rfloor$$

Thus, we conclude that the decryption algorithm returns the addition mod 2 of the bitstrings. Figure 3 illustrates a high-level description of how errors accumulate at the end of the decryption process. □

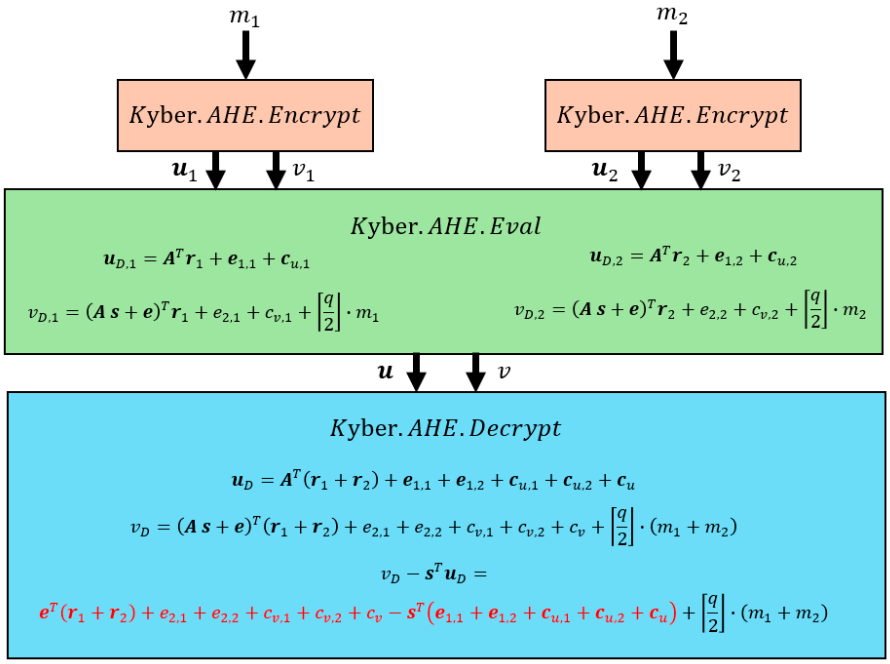


Figure 3. This illustration shows a schematic view of how errors grow at each stage. They are indicated in red.

For the sake of completeness, Theorem 2 states that subtraction mod 2 can be used in an encrypted domain to perform XOR on bit strings.

Theorem 2. *Let* $\boldsymbol{A} \in R_q^{kxk}$, $\boldsymbol{s}$, $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{e}$, $\boldsymbol{e}_{1,1}$, $\boldsymbol{e}_{1,2} \in R_q^k$, $e_{2,1}$, $e_{2,2} \in R_q$ *and let the messages* m_1, $m_2 \in R_q$ *with coefficients in* $\{0, 1\}$. *It is assumed that* $q \equiv 1\ mod\ 4$. *Define:*

- $\boldsymbol{t}$, ρ, *and* $\boldsymbol{s}$ *as the output* $(\boldsymbol{t}, \rho)$ *and* $\boldsymbol{s}$ *of Algorithm 1.*
- $(\boldsymbol{u}_1, v_1)$ *as the output of Algorithm 2 applied to* m_1 *using* $\boldsymbol{t}$ *and* ρ. *Similarly,* $(\boldsymbol{u}_2, v_2)$ *as the output of Algorithm 2 is applied to* m_2 *using the same* $\boldsymbol{t}$ *and* ρ.
- $(\boldsymbol{u}, v)$ *as the output of Algorithm 5 applied to* $(\boldsymbol{u}_1, v_1)$ *and* $(\boldsymbol{u}_2, v_2)$ *using* $\boldsymbol{t}$, *it is, the coefficients of the polynomials in the ciphertexts are subtracted.*
- $\boldsymbol{c}_{u,1}$, $\boldsymbol{c}_{u,2}$, $c_{v,1}$, $c_{v,1}$ *as the compression errors of* $(\boldsymbol{u}_1, v_1)$ *and* $(\boldsymbol{u}_2, v_2)$ *introduced in Algorithm 2.*
- $\boldsymbol{c}_u$, c_v *as the compression errors of* $(\boldsymbol{u}, v)$ *introduced in Algorithm 5.*
- m' *as the output of Algorithm 3 applied to* $(\boldsymbol{u}, v)$ *using* $\boldsymbol{s}$.

Algorithm 5. $KYBER.AHE.Eval(c_1 = (\boldsymbol{u}_1,\ v_1), c_2 = (\boldsymbol{u}_2,\ v_2))$

1: $\boldsymbol{u_{D,1}} \coloneqq\ Decompress_q(\boldsymbol{u}_1, d_u)$
2: $\boldsymbol{u_{D,2}} \coloneqq\ Decompress_q(\boldsymbol{u}_2, d_u)$
3: $v_{D,1} \coloneqq\ Decompress_q(v_1, d_v)$
4: $v_{D,2} \coloneqq\ Decompress_q(v_2, d_v)$
5: $\boldsymbol{u_{sub}} \coloneqq \boldsymbol{u_{D,1}} - \boldsymbol{u_{D,2}}\ mod^+\ q$
6: $v_{sub} \coloneqq v_{D,1} - v_{D,2}\ mod^+\ q$
7: $\boldsymbol{u} \coloneqq Compress_q(\boldsymbol{u_{sub}}, d_u)$
8: $v \coloneqq Compress_q(v_{sub}, d_v)$
9: ***return*** $(\boldsymbol{u}, v)$

If the following is true,

$$\left\| \boldsymbol{e}^T(\boldsymbol{r}_1 - \boldsymbol{r}_2) + e_{2,1} - e_{2,2} + c_{v,1} - c_{v,2} + c_v - \boldsymbol{s}^T(\boldsymbol{e}_{1,1} - \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} - \boldsymbol{c}_{u,2} + \boldsymbol{c}_u) \right\|_\infty < \left\lceil \frac{q}{4} \right\rfloor, \tag{24}$$

then, Algorithm 5 returns $m' \in R_q$ with coefficients being $m_1 \oplus m_2$, it is, the XOR between the coefficients of m_1 and m_2.

The proof of Theorem 2 has the same development as proof of Theorem 1.

In biometric template protection, normally only two values are compared as will be discussed in the following Section. However, for completeness, we include the following corollary to extend the scheme to have depth N.

Corollary 1. *Let* $\boldsymbol{A} \in R_q^{kxk}$, $\boldsymbol{s}, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_N, \boldsymbol{e}, \boldsymbol{e}_{1,1}, \ldots, \boldsymbol{e}_{1,N} \in R^k$, $e_{2,1}, \ldots, e_{2,N} \in R$. *Let* N *messages* $m_1, \ldots, m_N \in R_q$ *with coefficients in* $\{0, 1\}$. *It is assumed that* $q \equiv 1\ mod\ 4$. *Define:*

- $\boldsymbol{t}, \rho$, *and* $\boldsymbol{s}$ *as the output* $(\boldsymbol{t}, \rho)$ *and* $\boldsymbol{s}$ *of Algorithm 1.*
- $(\boldsymbol{u}_i, v_i)$ *as the output of Algorithm 2 applied to the* i*-th message* m_i *using* $\boldsymbol{t}$ *and* ρ.
- $(\boldsymbol{u}, v)$ *as the output of Algorithm 4 extended to the same operations on* i *ciphertexts* $(\boldsymbol{u}_1, v_1), \ldots, (\boldsymbol{u}_N, v_N)$ *using* $\boldsymbol{t}$ *and* ρ.
- $\boldsymbol{c}_{u,1}, \ldots, \boldsymbol{c}_{u,N}, c_{v,1}, \ldots, c_{v,N}$ *as the compression errors introduced in Algorithm 2.*
- $\boldsymbol{c}_u, c_v$ *as the compression errors introduced in Algorithm 4 to obtain* $(\boldsymbol{u}, v)$.
- m *as the output of Algorithm 3 applied to* $(\boldsymbol{u}, v)$.

If the following is true,

$$\left\| \boldsymbol{e}^T \left(\sum_{i=1}^{N} \boldsymbol{r_i} \right) + \sum_{i=1}^{N} e_{2,i} + \sum_{i=1}^{N} c_{v,i} + c_v - \boldsymbol{s}^T \left(\sum_{i=1}^{N} \boldsymbol{e}_{1,i} + \sum_{i=1}^{N} \boldsymbol{c}_{u,i} + \boldsymbol{c}_u \right) \right\|_\infty < \left\lceil \frac{q}{4} \right\rfloor \tag{25}$$

then, Algorithm 3 returns $m \in R_q$ with coefficients being $m_1 + \ldots + m_N\ mod\ 2 = m_1 \oplus \ldots \oplus m_N$.

The proof of this corollary is trivial following Theorem 1.

3.2. Security (IND-CPA)

The Kyber AHE scheme is tightly IND-CPA secure under the hardness of the Module-LWE problem in the Random Oracle Model (ROM). We argue this with the following theorem.

Theorem 3 (Security of Kyber AHE)**.** *Suppose Sam is a random oracle. For any adversary A, there exist adversaries B and C with roughly the same running time as that of A such that* $\boldsymbol{Adv}^{CPA}_{Kyber.AHE}(A) \leq 2 \cdot \boldsymbol{Adv}^{MLWE}_{k+1,k,\eta}(B) + \boldsymbol{Adv}^{prf}_{PRF}(C)$.

The proof is straightforward since Kyber AHE is based on Kyber PKE. Then, $\boldsymbol{Adv}^{CPA}_{Kyber.AHE}(\mathrm{A}) = \boldsymbol{Adv}^{CPA}_{Kyber.PKE}(\mathrm{A})$. The proof of the IND-CPA security of Kyber PKE is based on the fact that, under the Module-LWE assumption, the public key and a ciphertext are pseudorandom. This proof, along with a more comprehensive analysis of attacks against it, can be found in [24,41].

3.3. The Error Bound and Parameter Selection

Since Kyber PKE uses probabilistic algorithms, the decryption in Theorem 1 leads to the addition modulo 2 of the messages m_1 and m_2 with probability $1 - \delta_h$, that is, there is a probability δ_h that the decryption does not result in the XOR of the messages. In order to obtain the probability of decryption failure, we will use the following method:

First, let us define the distribution ψ_d^k over R as follows:

$$1.\ \boldsymbol{y} \leftarrow R^k$$
$$2.\ return\ \left(\boldsymbol{y} - Decompress_q\left(Compress_q(\boldsymbol{y}, d), d\right)\right)\ mod^{\pm}\ q$$

Similarly, the distribution ψ_d is defined as follows:

$$1.\ y \leftarrow R$$
$$2.\ return\ \left(y - Decompress_q\left(Compress_q(y, d), d\right)\right)\ mod^{\pm}\ q$$

Assuming that compression errors are distributed as ψ_d^k, that is, they are pseudorandom due to the hardness of the Module-LWE problem, thus $\boldsymbol{c}_{u,1}, \boldsymbol{c}_{u,2}, \boldsymbol{c}_u \in \psi_d^k$, $c_{v,1}$, $c_{v,2}$ $c_v \in \psi_d$. Then, taking into consideration Theorem 1, δ_h can be computed as follows:

$$\delta_h = P\left[\left\{\begin{array}{c} s, \boldsymbol{r}_1, \boldsymbol{r}_2, e, \boldsymbol{e}_{1,1}, \boldsymbol{e}_{1,2} \in \beta_\eta^k; \\ e_{2,1}, e_{2,2} \in \beta_\eta; \\ \boldsymbol{c}_{u,1}, \boldsymbol{c}_{u,2}, \boldsymbol{c}_u \in \psi_d^k; \\ c_{v,1}, c_{v,2}\ c_v \in \psi_d; \\ \left\| \begin{array}{c} \boldsymbol{e}^T(\boldsymbol{r}_1 + \boldsymbol{r}_2) + \boldsymbol{e}_{2,1} + \boldsymbol{e}_{2,2} + c_{v,1} + c_{v,2} + c_v \\ -\boldsymbol{s}^T(\boldsymbol{e}_{1,1} + \boldsymbol{e}_{1,2} + \boldsymbol{c}_{u,1} + \boldsymbol{c}_{u,2} + \boldsymbol{c}_u) \end{array} \right\|_\infty < \lceil \frac{q}{4} \rfloor \end{array}\right\}\right] \tag{26}$$

This method is very similar to that used by Kyber's authors in [41] to compute the failure probability, whose results are included in the official submission to the NIST competition in [24] and in the standardization in FIPS 203 [25].

Based on the script in Python for Kyber PKE found in [42], we have coded a script to calculate the failure probability and security of Kyber AHE depending on the parameters used (specifically, we used Python version 3.12.1). Our code can be found at https://github.com/RobertoRomanCrypto/Kyber-AHE-Failure (accessed on 15 August 2025). In Table 1, we show the failure probability of Kyber AHE using the same parameters as Kyber PKE used in Kyber KEM (and ML-KEM of FIPS203), along with the inverse of the failure probability. These results are for an application that requires a depth of two. We show also the security level (SL) of each instance, defined as in the Call for proposals of the NIST post-quantum competitions [5]. The parameter η_1 defines the noise of $\boldsymbol{s}$, $\boldsymbol{e}$, $\boldsymbol{r}_1$, and $\boldsymbol{r}_2$. The parameter η_2 defines the noise of $\boldsymbol{e}_1$ and e_2. In the case of Kyber768 and Kyber1024, η_1 and η_2 are equal because they are the values of η indicated in the definition of the PKE algorithms.

Table 1. Failure probabilities δ_h depending on the Kyber instance. Concrete parameters considered are shown. The results are for a depth of 2.

	SL	k	η_1	η_2	(d_u,d_v)	δ_h	$1/\delta_h$
Kyber512	I	2	3	2	(10, 4)	2^{-39}	$\approx 5.49 \cdot 10^{11}$
Kyber768	III	3	2	2	(10, 4)	2^{-45}	$\approx 3.51 \cdot 10^{13}$
Kyber1024	V	4	2	2	(11, 5)	2^{-68}	$\approx 2.95 \cdot 10^{20}$
-	III	3	2	2	(12, 6)	2^{-135}	$\approx 4.36 \cdot 10^{40}$

We can see that the failure probabilities are very low and from a practical point of view are sufficiently low for a biometric template protection application. In any case, since these probabilities are not extremely low (e.g., 2^{-128}), one may ask if they are sufficiently low to be secure. At time of writing, the works that exploit the approximate correctness of the homomorphic encryption scheme, for a concrete setting where the adversary is passive, can control the input messages of the system and have access to a decryption oracle [43,44]. We argue that these attacks are not a concern in a biometric template protection setting where servers are passive honest-but-curious and the user does not have access to other protected templates. However, with the included Python code, lower failure probabilities can be achieved in case that IND-CPA security is desired. For example, we can use the same parameters of Kyber768 with $(d_u, d_v) = (12, 6)$ to achieve a failure probability of 2^{-135} and maintaining the security level III.

Table 2 shows the failure probability results when the Kyber AHE scheme with a depth of 3 is required. This can be used, for example, for the MPC scheme proposed in [34], in which Kyber is combined with garbled circuits. As can be seen, the error growth is still manageable, since the inverse of the failure probability is more than eight million with the worst-case parameter set. The table shows alternative parameter sets for a lower failure probability.

Table 2. Failure probabilities δ_h depending on the Kyber instance. Concrete parameters considered are shown. The results are for a depth of 3.

	SL	k	η_1	η_2	(d_u,d_v)	δ_h	$1/\delta_h$
Kyber512	I	2	3	2	(10, 4)	2^{-23}	$\approx 8.39 \cdot 10^{6}$
Kyber768	III	3	2	2	(10, 4)	2^{-26}	$\approx 6.71 \cdot 10^{7}$
Kyber1024	V	4	2	2	(11, 5)	2^{-43}	$\approx 8.80 \cdot 10^{12}$
-	I	2	3	2	(11, 5)	2^{-47}	$\approx 1.41 \cdot 10^{14}$
-	III	3	2	2	(11, 5)	2^{-53}	$\approx 9.01 \cdot 10^{15}$

The fact that Kyber AHE can use the same parameters as Kyber PKE makes the scheme very easy to implement.

3.4. Protection Against Side-Channel Attacks

The side-channel analysis of Kyber is an active area of research. As a result, various side-channel attacks have been identified against Kyber, including those in constant-time implementations [45–47]. Several works propose masked versions of Kyber to deal with side-channel attacks, which can also be applied to Kyber AHE. One study proposes a first-order Kyber masked implementation optimized for an ARM Cortex-M4 processor [48]. Another study suggests using a masked hardware/software co-design. Other studies propose solutions to mask specific operations. To mask the compression function, the work in [49] proposes a bit-sliced binary search of the 12-bit Boolean version of an arithmetically

shifted coefficient share to determine if the share is greater than or equal to $\lfloor \frac{q}{2} \rfloor$. The work in [50] also proposes two additional masking methods for the compression function: a double and check algorithm that adds a level of obfuscation and the integration of a look-up table (LUT) that uses one arithmetic-to-Boolean conversion for each share. To mask the symmetric operations used in Kyber, the method used proposed in [51] is often recommended. To mask the central binomial distribution sampler, the approach shown in [52] can be used. Other countermeasures used to deal with side-channel attacks include adding noise (mainly Gaussian noise), introducing random delays during the execution of the encryption and decryption algorithms, and introducing clock instability (clock jitter) [53].

4. Application to Biometric Template Protection

In the following, we present the experimental results of Kyber AHE for the setup described in Section 2.1 and illustrated in Figure 1. The experiments were conducted using a commercial laptop equipped with an 11th generation Intel® Core™ i7-11800H @ 2.30 GHz processor and 16.0 GB of RAM. The operating system used was Windows 10. Visual Studio Code version 1.102.3 was used as an integrated development environment (IDE). Execution times were averaged from 1000 test runs.

Our implementation is based on the Kyber C implementation found on GitHub at https://github.com/pq-crystals/kyber/tree/main/ref (accessed on 17 May 2025). Specifically, it is the *reference (ref)* implementation. The *indcpa.c* file contains the key generation (*indcpa_keypair*), encryption (*indcpa_enc*), and decryption (*indcpa_dec*) functions. We implemented the evaluation function as *indcpa_eval* in the same file. Decompression and compression are performed using the *poly_decompress*, *polyvec_decompress*, *poly_compress*, and *polyvec_compress* functions, which are implemented in the *poly.c* and *polyvec.c* files. Addition modulo q is performed using the *poly_add* and *polyvec_add* functions. Note that the modulus q has a value of 3329. We encourage readers to refer to [24] to see the exact implementations of the Kyber algorithms. The implementation of Kyber AHE can be found at https://github.com/RobertoRomanCrypto/Kyber-AHE (accessed on 15 August 2025).

We use 256-byte iris templates, which is the size used for the popular IrisCodes [54]. Thus, if we want to encrypt an iris template, we should use 8 ciphertexts because Kyber AHE encrypts 32 bytes at a time. Figure 4 shows how the protected data are managed by the Client, the Database Server, and the Matching Server.

We note that decryption failures in Kyber AHE are extremely rare considering the results in Table 1. Thus, the recognition accuracy of IrisCodes before and after encryption is the same. The analysis of the security and performance recognition of IrisCodes has been extensively studied in the literature [55–59]. Therefore, in this paper, we focus on the performance in terms of sizes and execution times of Kyber AHE to protect the iris features. We present results for three NIST security levels I, III, and V corresponding to Kyber512, Kyber768, and Kyber1024, respectively. It is shown in [24] that they have a quantum core-SVP hardness of 107, 165, and 232 bits, respectively.

Table 3 shows the execution times to carry out each operation as long as the size of the encrypted iris features. The execution times shown are those required for the Client to encrypt an iris feature, for the Database Server to evaluate an addition modulo 2 in the encrypted domain using the Kyber AHE evaluation algorithm, and for the Matching Server to decrypt the result.

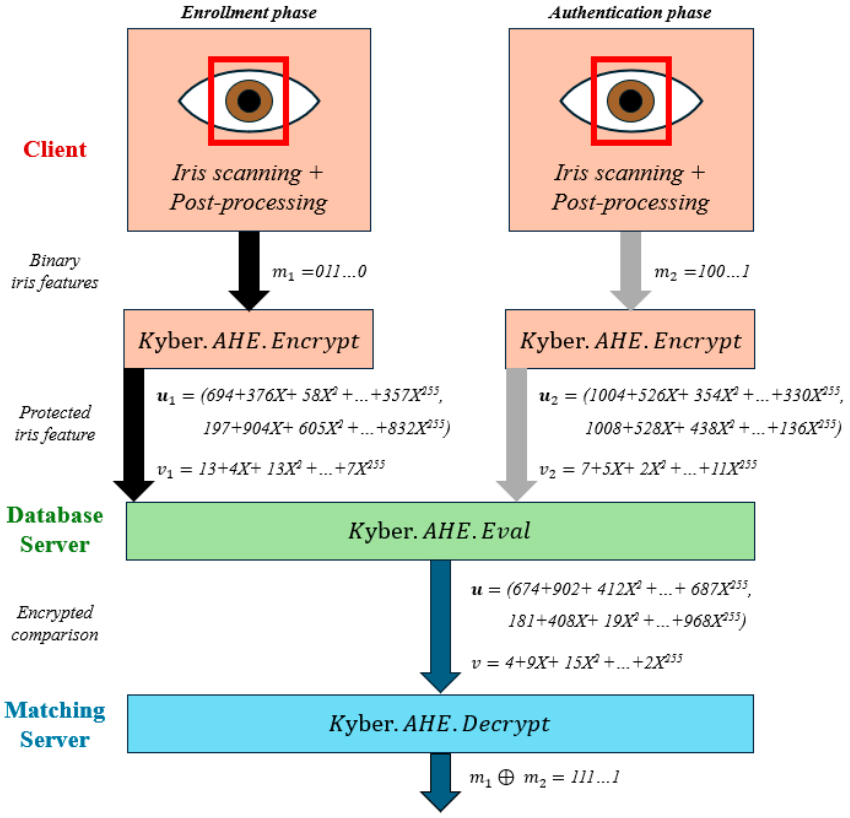


Figure 4. Illustration of protected data managed by the Client, the Database and the Matching Server.

Table 3. Performance results in terms of execution times. The results from this work are in grey.

	Execution Times (in ms)	
	Encrypt	**Evaluate + Decrypt**
Kyber512 (256 bytes)	0.755	0.096 + 0.259
Kyber768 (256 bytes)	1.25	0.131 + 0.343
Kyber1024 (256 bytes)	1.73	0.161 + 0.415
Id. latt. * (256 bytes) [31]	19.89	18.10 + 9.08
CKKS ** (512 Floating Point values) [30]	~6	~3391
BFV ** (512 integer values) [30]	~76	~168

* In an Intel Xeon X3480 at 3.07 GHz with 16 GB of memory. ** In an Intel Core i7 at 2.7 GHz with 16 GB of memory.

The sizes of an encrypted iris feature are shown in Figure 5. They are not very large. An encrypted database of 100,000 individuals will require between 586 MB and 1197 MB using Kyber AHE for iris encryption, which is not a very large number for a database server. Also, the times are very low, even using the reference Kyber implementation that is not optimized for the AVX2 instruction set. The size of the public key is 800, 1184, and 1568 bytes for security levels I, III, and V, respectively. The size of the secret key is 32 bytes.

Table 3 also compares our solution with others from the literature. The authors of [30] claim they have used a bit security of 128 bits for CKKS and BFV. The work in [31] claims a security of 80 bits against BKZ 2.0 [60] with a certain margin. The format of the template is shown in parentheses. Clearly, our solution outperforms the homomorphic solutions in [31] that encrypt binary biometric strings. Additionally, compared with the solutions using CKKS and BFV in [30], it can be concluded that using binary instead of floating-point and integer format templates is important for achieving low-cost implementations with simpler operations.

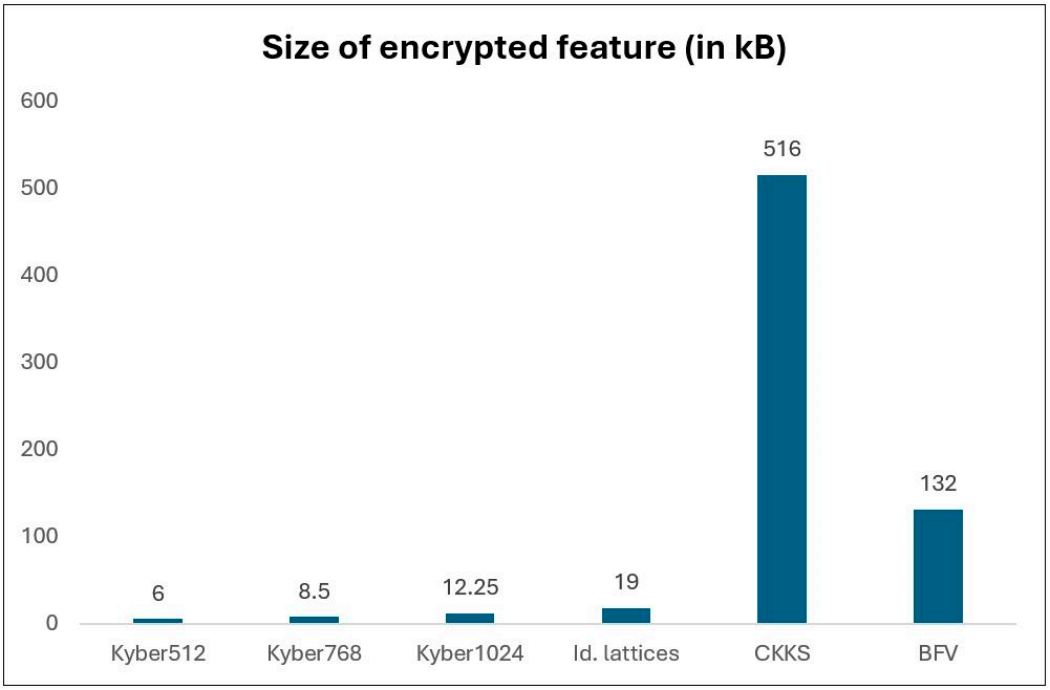


Figure 5. Comparison of the size of the encrypted biometric feature in kilobytes between our work and that of [30], which uses CKKS and BFV, as well as that of [31], which uses ideal lattices. For our proposal using Kyber AHE, we used a 256-byte biometric feature. The proposal using ideal lattices also uses a 256-byte biometric feature. The CKKS proposal uses 512 floating-point values, while the BFV proposal uses 512 integer values.

The execution times shown in Table 3 do not consider countermeasures to prevent leakage of the XOR between the reference and fresh biometric features. As mentioned in Section 2.1, potential countermeasures include adding a cancellable biometric method before encrypting the biometric feature and using an MPC solution. Other works from the literature can provide a qualitative idea of the overhead cost of implementing one of these schemes.

The work in [33] uses a hybrid method combining three hashing methods and homomorphic encryption with the BFV scheme. For example, results from a system with an Intel® Core™ i7-7700K CPU @ 4.20 GHz processor demonstrate that the Biohashing operation requires 15.08 ms to protect a biometric feature obtained from an ArcFace face recognition model, accounting for 4.36% of the total execution time of the hybrid scheme. Additionally, cancellable biometrics do not significantly increase bandwidth usage in their scheme. Thus, combining cancelable biometrics with the Kyber AHE scheme seems completely affordable.

On the other hand, the work in [34] combines homomorphic encryption and garbled circuits to protect facial biometric features from FaceNet. Only the verification decision is revealed to the authentication server. The results show that the oblivious transfer adds nearly 0.5 s of overhead to the baseline system. While this overhead is significant, the solution is feasible, especially considering that the oblivious transfer is performed between two servers. Also, their proposal is affordable in terms of communication costs.

5. Conclusions

In this work, we propose Kyber AHE, an additive homomorphic encryption scheme used to compute the addition modulo 2 of two binary strings in the encrypted domain. The scheme consists of three algorithms: Kyber.AHE.KeyGen, Ky-ber.AHE.Encrypt, Ky-ber.AHE.Eval, and Kyber.AHE.Decrypt. The addition modulo 2 can be used to compute the Hamming distance between two biometric features in binary format. This makes Kyber

AHE very suitable for the protection of biometric templates. An analysis of the failure probability shows that using the parameters of Kyber512, Kyber768, and Kyber1024 instances has a very low probability of decryption failure (smaller than 2^{-39} in the worst case). Consequently, the use of Kyber AHE with these instances does not degrade the recognition performance when used in a biometric template protection scheme. Experimental results for the protection of a 256-byte iris feature show that the size of the protected feature is relatively small (smaller than 12.25 kB in the worst case) and the execution times are also very low (shorter than 1.73 ms in the worst case of the encryption algorithm), which clearly outperform other homomorphic solutions reported in the literature.

Author Contributions: Conceptualization, R.R., R.A. and I.B.; methodology, R.R., R.A. and I.B.; software, R.R.; validation, R.R., R.A. and I.B.; formal analysis, R.R., R.A. and I.B.; investigation, R.R., R.A. and I.B.; data curation, R.R.; writing—original draft preparation, R.R., R.A. and I.B.; writing—review and editing, R.R., R.A. and I.B.; supervision, R.A. and I.B.; project administration, R.A. and I.B.; funding acquisition, R.A. and I.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was conducted thanks to Grants PDC2023–145873-I00, CPP2022–009796, and PID2023-150809OB-I00 funded by MICIU/AEI/10.13039/501100011033 and the European Union NextGenerationEU/PRTR, thanks to the LICORICE Project with Grant Agreement No. 101168311 under the EU Horizon Europe, and thanks to the grant USECHIP (TSI-069100-2023-001) funded by the Secretary of State for Telecommunications and Digital Infrastructure, Ministry for Digital Transformation and Civil Service and by the European Union—Next Generation EU/PRTR. The work of Roberto Román was supported by VI Plan Propio de Investigación y Transferencia, Universidad de Sevilla.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author(s).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Aydin, F.; Karabulut, E.; Seetal, P.; Alkim, E.; Aysu, A. RevEAL: Single-trace side-channel leakage of the SEAL homomorphic encryption library. *IACR Cryptol. ePrint Arch.* **2022**, *2022*, 204.
2. Aydin, F.; Aysu, A. Exposing side-channel leakage of SEAL homomorphic encryption library. In Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security (ASHES'22), Los Angeles, CA, USA, 11 November 2022; pp. 95–100. [CrossRef]
3. Cheng, W.; Danger, J.-L.; Guilley, S.; Huang, F.; Bel Korchi, A.; Rioul, O. Cache-timing attack on the SEAL homomorphic encryption library. In Proceedings of the 11th International Workshop on Security Proofs for Embedded Systems (PROOFS 2022), Leuven, Belgium, 22 September 2022; pp. 1–15.
4. Aydin, F.; Aysu, A. RevEAL: Leaking secrets in homomorphic encryption with side-channel attacks. *IACR Cryptol. ePrint Arch.* **2023**, *2023*, 1128.
5. Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process. 2016. Available online: https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf (accessed on 25 August 2025).
6. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (General Data Protection Regulation) (Text with EEA Relevance). 2016. Available online: https://eur-lex.europa.eu/eli/reg/2016/679/oj/eng (accessed on 25 August 2025).
7. Gomez-Barrero, M.; Maiorana, E.; Galbally, J.; Campisi, P.; Fierrez, J. Multi-biometric template protection based on homomorphic encryption. *Pattern Recognit.* **2017**, *67*, 149–163. [CrossRef]
8. Drozdowski, P.; Buchmann, N.; Rathgeb, R.; Margraf, M.; Busch, C. On the application of homomorphic encryption to face identification. In Proceedings of the 2019 International Conference on Biometrics (ICB 2019), Crete, Greece, 4–7 June 2019; pp. 173–180.
9. Yang, W.; Wang, W.; Cui, H.; Tang, Z.; Li, Y. A review of homomorphic encryption for privacy-preserving biometrics. *Sensors* **2023**, *23*, 3566. [CrossRef]

10. Armknecht, F.; Boyd, C.; Carr, C.; Gjøsteen, K.; Jäschke, A.; Reuter, C.-A.; Strand, M. A guide to fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 1192.
11. Fan, J.; Vercauteren, F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2012**, *2012*, 144.
12. Pflug, A.; Rathgeb, C.; Scherhag, U.; Busch, C. Binarization of spectral histogram models: An application to efficient biometric identification. In Proceedings of the 2015 IEEE 2nd International Conference on Cybernetics (CYBCONF), Gdynia, Poland, 24–26 June 2015; pp. 501–506. [CrossRef]
13. Hamouda, E.; Yuan, X.; Ouda, O.; Hamza, T. Secure and efficient biometric-data binarization using multi-objective optimization. *Int. J. Comput. Intell. Syst.* **2015**, *8*, 1116–1127. [CrossRef]
14. Lim, M.-H.; Teoh, A.B.J. A novel encoding scheme for effective biometric discretization: Linearly separable subcode. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 300–313. [CrossRef]
15. Bloemen, R.; Gillespie, B.; Kales, D.; Sippl, P.; Walch, R. Large-scale MPC: Scaling private iris code uniqueness checks to millions of users. *IACR Cryptol. ePrint Arch.* **2024**, *2024*, 705.
16. Bauspieß, P.; Vad, L.; Myrekrok, H.; Costache, A.; Kolberg, J.; Rathgeb, C.; Busch, C. On the feasibility of fully homomorphic encryption of minutiae-based fingerprint representations. In Proceedings of the 9th International Conference on Information Systems Security and Privacy (ICISSP 2023), Lisbon, Portugal, 22–24 February 2023; pp. 462–470.
17. Kolberg, J.; Bauspieß, P.; Gomez-Barrero, M.; Rathgeb, C.; Dürmuth, M.; Busch, C. Template protection based on homomorphic encryption: Computationally efficient application to iris-biometric verification and identification. In Proceedings of the 2019 IEEE International Workshop on Information Forensics and Security (WIFS 2019), Delf, The Netherlands, 9–12 December 2019; pp. 1–6.
18. Hoffstein, J.; Pipher, J.; Silverman, J.H.; Rathgeb, C.; Busch, C. NTRU: A ring-based public key cryptosystem. In Proceedings of the Third International Symposium on Algorithmic Number Theory (ANTS 1998), Portland, OR, USA, 21–25 June 1998; pp. 1–22.
19. Stehlé, D.; Steinfeld, R. Making NTRU as secure as worst-case problems over ideal lattices. In Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques—EUROCRYPT 2011, Tallin, Estonia, 15–19 May 2011; pp. 1–20.
20. Albrecht, M.; Bai, S.; Ducas, L. A subfield lattice attack on overstretched NTRU assumptions. In Proceedings of the 36th Annual International Cryptology Conference on Advances in Cryptology—CRYPTO 2016, Santa Barbara, CA, USA, 14–18 August 2016; pp. 153–178.
21. Cramer, R.; Ducas, L.; Peikert, C.; Regev, O. Recovering Short Generators of Principal Ideals in Cyclotomic Rings. In Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques—EUROCRYPT 2016, Vienna, Austria, 8–12 May 2016; pp. 1–23.
22. NTRU. Algorithm Specifications And Supporting Documentation. Round 3 Nist Submission. 2020. Available online: https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions (accessed on 25 April 2025).
23. NIST Post-Quantum Cryptography. Selected Algorithms. Available online: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms (accessed on 25 August 2025).
24. CRYSTALS-Kyber. Algorithm Specifications and Supporting Documentation (version 3.01) Round 4 NIST Submission. 2021. Available online: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210131.pdf (accessed on 25 August 2025).
25. National Institute of Standards and Technology: Module-lattice-based key encapsulation mechanism standard. In *Federal Information Processing Standards Publication (FIPS) NIST FIPS 203 IPD*; Department of Commerce: Washington, DC, USA, 2023.
26. Schwabe, P.; Douglas, S.; Wiggers, T. Post-quantum TLS without handshake signatures. *IACR Cryptol. ePrint Arch.* **2020**, *2020*, 534.
27. Alnahawi, N.; Müller, J.; Oupický, J.; Wiesmaier, A. A comprehensive survey on post-quantum TLS. *IACR Commun. Cryptol.* **2024**, *1*, 2. [CrossRef]
28. Román, R.; Arjona, R.; López-González, P.; Baturone, I. A quantum-resistant face template protection scheme using Kyber and Saber public key encryption algorithms. In Proceedings of the 21th International Conference of the Biometrics Special Interest Group (BIOSIG 2022), Darmstadt, Germany, 15–16 September 2022; pp. 1–5.
29. Franco, C.; Arjona, R.; Baturone, I. A cloud-based multifactor authentication scheme using post-quantum cryptography and trusted execution environments. In Proceedings of the Availability, Reliability and Security, ARES 2025 EU Projects Symposium Workshops, Ghent, Belgium, 11–14 August 2025; pp. 217–234.
30. Kolberg, J.; Drozdowski, P.; Gomez-Barrero, M.; Rathgeb, C.; Busch, C. Efficiency analysis of post-quantum-secure face template protection schemes based on homomorphic encryption. In Proceedings of the 19th International Conference of the Biometrics Special Interest Group (BIOSIG 2020), Darmstadt, Germany, 17–18 September 2020; pp. 1–4.
31. Yasuda, M.; Shimoyama, T.; Kogure, J.; Yokoyama, K.; Koshiba, T. Packed Homomorphic Encryption Based on Ideal Lattices and Its Application to Biometrics. In Proceedings of the 2013 CD-ARES: International Conference on Availability, Reliability, and Security—CD-ARES 2013, Regensburg, Germany, 2–6 September 2013; pp. 55–74.

32. Abidin, A.; Aly, A.; Rúa, E.A.; Mitrokotsa, A. Efficient Verifiable Computation of XOR for Biometric Authentication. In Proceedings of the 15th International Conference on Cryptology and Network Security—CANS 2016, Milan, Italy, 14–16 November 2016; pp. 284–298.
33. Shahreza, H.O.; Rathgeb, C.; Osorio-Roig, D.; Krivokuća Hahn, V.; Marcel, S.; Busch, C. Hybrid Protection of Biometric Templates by Combining Homomorphic Encryption and Cancelable Biometrics. In Proceedings of the 2022 IEEE International Joint Conference on Biometrics (IJCB 2022), Abu Dhabi, United Arabian Emirates, 10–13 October 2022; pp. 1–10.
34. Arjona, R.; Franco, C.; Román, R.; Baturone, I. Combining CRYSTALS-Kyber Homomorphic Encryption with Garbled Circuits for Biometric Authentication. In Proceedings of the 23th International Conference of the Biometrics Special Interest Group (BIOSIG 2020), Darmstadt, Germany, 25–27 September 2024; pp. 1–5.
35. Xu, Z.; Pemberton, O.; Roy, S.S.; Oswald, D.; Yao, W.; Zheng, Z. Magnifying side-channel leakage of lattice-based cryptosystems with chosen ciphertexts: The case study of kyber. *IEEE Trans. Comput.* **2022**, *71*, 2163–2176. [CrossRef]
36. Reparaz, O.; de Clercq, R.; Roy, S.S.; Vercauteren, F.; Verbauwhede, I. Additively homomorphic ring-LWE masking. In Proceedings of the 7th International Conference on Post-Quantum Cryptography—PQCrypto 2016, Fukuoka, Japan, 24–26 February 2016; pp. 233–244. [CrossRef]
37. Primas, R.; Pessl, P.; Mangard, S. Single-trace side-channel attacks on masked lattice-based encryption multimodal IoT. *IACR Cryptol. ePrint Arch.* **2022**, *2022*, 594.
38. Román, R.; Arjona, R.; Baturone, I. A quantum-safe authentication scheme for IoT devices using homomorphic encryption and weak physical unclonable functions with no helper data. *Internet Things* **2024**, *28*, 101389. [CrossRef]
39. Román, R.; Arjona, R.; Baturone, I. Multimodal IoT device authentication using behavioral and physical unclonable functions and kyber public key encryption. In Proceedings of the 21st International Conference on Synthesis, Modeling, Analysis and Simulation Methods, and Applications to Circuits Design—SMACD 2025, Istanbul, Turkiye, 7–10 July 2025; pp. 1–5.
40. Kreuzer, K. Verification of correctness and security properties for CRYSTALS-KYBER. In Proceedings of the IEEE 37th Computer Security Foundations Symposium (CSF 2024), Enschede, The Netherlands, 8–12 July 2024; pp. 511–526.
41. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.-M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS—Kyber: A CCA-secure module-lattice-based KEM. In Proceedings of the 3rd IEEE European Symposium on Security and Privacy (EuroS&P 2018), London, UK, 24–26 April 2018; pp. 353–367.
42. Security Estimation Scripts for Kyber and Dilithium. Available online: https://github.com/pq-crystals/security-estimates (accessed on 25 August 2025).
43. Li, B.; Micciancio, D. On the Security of Homomorphic Encryption on Approximate Numbers. In Proceedings of the 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques—EUROCRYPT 2016, Vienna, Austria, 8–12 May 2016; pp. 1–27. [CrossRef]
44. Cheon, J.-H.; Choe, H.; Passelègue, A.; Stehlé, D.; Suvanto, E. Attacks Against the IND-CPA^D Security of Exact FHE Schemes. In Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS 2024), Salt Lake City, UT, USA, 14–18 October 2024; pp. 2505–2519.
45. Ravi, P.; Roy, S.S.; Chattopadhyay, A.; Bhasin, S. Generic side-channel attacks on CCA-secure lattice-based PKE and KEMs. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 948. [CrossRef]
46. Ravi, P.; Bhasin, S.; Roy, S.S.; Chattopadhyay, A. Drop by drop you break the rock- exploiting generic vulnerabilities in lattice-based PKE/KEMs using EM-based physical attacks. *IACR Cryptol. ePrint Arch.* **2020**, *2020*, 549.
47. Pessl, P.; Primas, R. More practical single-trace attacks on the number theoretic transform. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 795.
48. Heinz, D.; Kannwischer, M.J.; Land, G.; Pöppelmann, T.; Schwabe, P.; Sprenkels, A. First-Order Masked Kyber on ARM Cortex-M4. *IACR Cryptol. ePrint Arch.* **2022**, *2022*, 058.
49. Bos, J.W.; Gourjon, M.; Renes, J.; Schneider, T.; van Vredendaal, C. Masking kyber: First- and higher-order implementations. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 173–214. [CrossRef]
50. Özeren, S.; Yayla, O. Methods for masking CRYSTALS-kyber against side-channel attacks. *IACR Cryptol. ePrint Arch.* **2023**, *2023*, 1340.
51. Barthe, G.; Belaïd, S.; Dupressoir, F.; Fouque, P.-A.; Grégoire, B.; Strub, P.-Y.; Zucchini, R. Strong non-interference and type-directed higher-order masking. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 506.
52. Schneider, T.; Paglialonga, C.; Oder, T.; Güneysu, T. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 910.
53. Huang, Z.; Wang, H.; Cao, B.; He, D.; Wang, J. A comprehensive side-channel leakage assessment of CRYSTALS-Kyber in IIoT. *Internet Things* **2024**, *27*, 101331. [CrossRef]
54. Daugman, J. Information Theory and the IrisCode. *IEEE Trans. Inf. Forensics Secur.* **2015**, *11*, 400–409. [CrossRef]
55. Daugman, J. High confidence visual recognition of persons by a test of statistical independence. *IEEE Trans. Pattern Anal. Mach. Intell.* **1993**, *15*, 1148–1161. [CrossRef]

56. Daugman, J. The importance of being random: Statistical principles of iris recognition. *Pattern Recognit.* **2003**, *36*, 279–291. [CrossRef]
57. Daugman, J. Probing the uniqueness and randomness of IrisCodes: Results from 200 billion iris pair comparisons. *Proc. IEEE* **2006**, *94*, 1927–1935. [CrossRef]
58. Kong, A.-W.-K.; Zhang, D.; Kamel, M.S. An analysis of IrisCode. *IEEE Trans. Image Process* **2010**, *19*, 522–532. [CrossRef]
59. Daugman, J. Statistical richness of visual phase information. *Int. J. Comput. Vis.* **2001**, *45*, 25–38. [CrossRef]
60. Chen, Y.; Nguyen, P.Q. BKZ 2.0: Better Lattice Security Estimates. In Proceedings of the Advances in Cryptology—ASIACRYPT 2011, Seoul, Republic of Korea, 4–8 December 2011; pp. 1–20.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.