sensors

MDPI

Article

Acceleration of Inner-Pairing Product Operation for Secure Biometric Verification †

Seong-Yun Jeon and Mun-Kyu Lee *

Department of Electrical and Computer Engineering, Inha University, Incheon 22212, Korea; roland.korea@gmail.com

* Correspondence: mklee@inha.ac.kr; Tel.: +82-32-860-7456

† This paper is an extended version of our paper published in the poster session of WISA 2020, "Seong-Yun Jeon; Mun-Kyu Lee. Poster: Acceleration of Pairing Product Operation Using Precomputation", where we have presented a Fixed-Q product method. In this paper, we provide a full description of this method, and present a facial verification application to demonstrate the feasibility of the proposed method. A part of this paper also appeared in the Master's Thesis, "Seong-Yun Jeon. Acceleration of Pairing Operation for Performance Improvement of Functional Encryption" in Korean

Abstract: With the recent advances in mobile technologies, biometric verification is being adopted in many smart devices as a means for authenticating their owners. As biometric data leakage may cause stringent privacy issues, many proposals have been offered to guarantee the security of stored biometric data, i.e., biometric template. One of the most promising solutions is the use of a remote server that stores the template in an encrypted form and performs a biometric comparison on the ciphertext domain, using recently proposed functional encryption (FE) techniques. However, the drawback of this approach is that considerable computation is required for the inner-pairing product operation used for the decryption procedure of the underlying FE, which is performed in the authentication phase. In this paper, we propose an enhanced method to accelerate the inner-pairing product computation and apply it to expedite the decryption operation of FE and for faster remote biometric verification. The following two important observations are the basis for our improvement—one of the two arguments for the decryption operation does not frequently change over authentication sessions, and we only need to evaluate the product of multiple pairings, rather than individual pairings. From the results of our experiments, the proposed method reduces the time required to compute an inner-pairing product by 30.7%, compared to the previous best method. With this improvement, the time required for biometric verification is expected to decrease by up to 10.0%, compared to a naive method.

Keywords: inner-pairing product; functional encryption; biometric verification

check for updates

Citation: Jeon, S.-Y.; Lee, M.-K. Acceleration of Inner-Pairing Product Operation for Secure Biometric Verification. *Sensors* **2021**, *21*, 2859. https://doi.org/10.3390/s21082859

Academic Editor: Ilsun You

Received: 10 February 2021
Accepted: 11 April 2021
Published: 19 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 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

Biometric recognition is the automated recognition of individuals based on their biological and behavioral characteristics. Biometric recognition has two types [1]: biometric identification and biometric verification. Biometric identification is a process that searches against a biometric enrolment database to find and return the biometric reference identifier(s) attributable to a single user. Conversely, biometric verification is a process that confirms a biometric claim through biometric comparison. In a biometric verification system, a user can make a biometric claim to a biometric characteristic examiner. When a user claims that he or she is the source of a specified biometric reference, the examiner may verify this claim by performing a biometric comparison. With the recent advances in mobile technologies, biometric verification is being adopted in many smart devices as a tool for authenticating their owners. This technique is used not only to unlock devices but also permit users to run security-critical applications, such as financial services [2].

For biometric verification, the biometric data of a user should be stored first during biometric enrolment. The stored biometric data are referred to as a biometric reference, and

they are stored using a data structure called a biometric template [1]. However, biometric characteristics are unique and unchangeable; this implies that leakage of these characteristics may cause more critical privacy issues than the compromise of existing passwords and personal identification numbers (PINs). Furthermore, users' biometric templates are often compromised, especially for mobile devices [3]. Therefore, a biometric verification method that ensures the security of the biometric template must be developed [4,5].

Several proposals are available in the literature to secure biometric templates without additional hardware support such as ARM Trust Zone [6] and Apple Secure Enclave [7]. First, there are methods to convert biometric data using noninvertible transforms, such as cancelable biometrics [8,9] and fuzzy commitment [10]. However, these methods have a problem of decreasing recognition accuracy owing to the conversion process. In addition, there are many cases where one-way transformations are analyzed and inverted successfully [11–13].

There are also biometric key generation techniques for biometric template protection [14–16]. Using these methods, a unique and high-entropy key can be generated from the user's biometric input on the fly. These methods have a very desirable property that the user's biometric template does not need to be stored in the device. However, to achieve both goals of providing a sufficient level of recognition rate and effectively generating biometric keys, these methods require additional tools. For example, the electrocardiogram-based biometric key generation method in [14] requires for helper data to be stored. According to [16], the use of helper data is not desirable. The electroencephalography (EEG)-based method in [15] requires high-bandwidth data with more than 60 channels. It was pointed out in [17] that in most commercially available EEG devices, less than 20 channels are provided. The fingerprint-based biometric key generation method in [16] requires an additional device such as a smart card. On the contrary, if a remote server is available, storing the biometric template on the remote server outside the device is another option for biometric verification without the need for additional tools or performance degradation [18–21]. In these methods, the server functions as a secure repository. This approach is also suitable when a user wants to be granted access to a particular remote service using an authentication server (e.g., online banking). However, this approach can raise another privacy issue because the remote server can learn the user's biometric characteristics. In other words, an honest-but-curious server may try to recover the user's biometric features using the user's stored template. In addition, attackers who intrude into the server may obtain the biometric features of the legitimate user. These features may be used to impersonate the legitimate user in another system [19–21]. Therefore, an encryption scheme that encrypts biometric data and makes the server examine the similarity in encrypted data is necessary for a privacy-preserving biometric verification.

Meanwhile, functional encryption (FE) is an encryption scheme in which possessing a secret key allows one to obtain only the result of $f(x)$ from a ciphertext $E(x)$, but not learn any information about the data x, where the secret key is related to the function f and the ciphertext $E(x)$ is the encryption of data x. Hence, FE is considered a suitable scheme for constructing a privacy-preserving biometric verification system [22–24]. As FE requires considerable computation, extensive research has been conducted to make FE more practical, particularly when the evaluated function is an inner product of a plaintext vector $\mathbf{x}$ with a vector $\mathbf{y}$ encoded in the function f [25–29]. Kim et al. proposed a practical inner product FE scheme with a function-hiding property, which implies that not only $\mathbf{x}$ but also $\mathbf{y}$ remain hidden [22]. They also provided a reference code implemented in Python and evaluated the required durations for main operations, such as key generation, encryption, and decryption. According to their measurement, the dominant operations took up to several seconds on a desktop PC, and this indicates that FE may guarantee a practical level of performance. Kim et al.'s scheme is very suitable as a base scheme for a privacy-preserving biometric verification system, but its decryption operation requires an inner-pairing product operation [22], which is its most time-consuming part. An inner-pairing product is the product of multiple pairings, and its inputs are two vectors

comprising points on a certain elliptic curve [30]. There are several optimization techniques for the inner-pairing product [31–34]. Scott suggested to share underlying operations for multiple pairings [31] and the validity of this approach was verified in [32,33]. Costello et al. proposed to apply precomputation to accelerate pairing operations [34].

In this paper, we propose an improved method for accelerating inner-pairing product computation by combining the shared computation techniques and precomputation. The experimental results indicate that the proposed method reduces the time required to compute an inner-pairing product by up to 30.7%. To cope with the situation where the memory is not sufficient to store all the precomputed data, we also propose an adaptive method that can adjust the number of elements to be precomputed and stored. According to our analysis, the performance of the proposed method can be fine-tuned adaptively according to the storage capacity. Furthermore, we demonstrate that the proposed method is suitable for application to a remote biometric verification system using FE, where one of the two inputs to the inner-pairing product operation is not frequently changed. Using the proposed method, we can assume that the performance of biometric verification will be enhanced by 9.0–10.0%.

1.1. Related Works

In 2005, Scott proposed three ideas to optimize the inner-pairing product computation for Tate pairing by sharing the common operations among the pairings [31]. In 2006, Granger et al. showed that Scott's method can also be applied to the inner-pairing product for Ate pairing [32]. In 2015, Zavattoni et al. proposed an optimized method to compute the products of optimal Ate pairings on the BN curve [33]. Meanwhile, in 2010, Costello et al. proposed a precomputation method to accelerate pairing computation when one argument of the pairing is fixed [34]. The more times the pairing is called, the more gain is obtained in the execution time, at the small expense of memory to store the precomputed elements. Recently, the pairing operations are being adopted as a crucial operation for many applications, such as privacy-preserving applications [35,36] and non-interactive zero-knowledge proofs [37,38].

There have been various research results in the literature for secure remote biometric verification [39–48]. In 2016, Cheon et al. proposed a homomorphic encryption (HE)-based protocol to support encrypted Hamming distance computation required for iris recognition [40]. In the same year, Im et al. [41] proposed an HE-based protocol to support encrypted Euclidean distance computation for palm print authentication [49]. In 2018, Gunasinghe and Bertino proposed a secure face verification protocol based on zero-knowledge proof of knowledge (ZKPK). To perform their protocol, a trusted execution environment (TEE) should be equipped on the device. In the same year, Droandi et al. [48] proposed a multi-party computation-based biometric matching protocol using SPDZ protocol [50]. Although their protocol was very fast, it required a high communication cost for preprocessing. In 2020, Im et al. proposed a secure face verification system that guarantees a real-time authentication performance on a smartphone [2]. They evaluated the performance of their system with two experiments; (1) an experiment involving 30 users in a real-world environment and (2) an experiment using the public face datasets CFP [51] and ORL [52]. In this paper, we consider the data presented in [2] as the criteria to evaluate the performance of the proposed method, since the result in [2] is one of the most up-to-date practical works in the literature of privacy-preserving biometric authentication. A more extensive survey of FE-based privacy-preserving applications can be found in [35].

1.2. Contributions

In this paper, we propose a method for accelerating inner-pairing product computation by combining the shared computation techniques and precomputation. To be precise, our contributions are as follows:

- We propose a new algorithm that accelerates inner-pairing product computation when one of the two input vectors are fixed. For each element in this fixed vector, we apply

the precomputation method that was originally proposed for a single pairing in [34]. In addition, we reduce the overall computational complexity by sharing overlapping operations for multiple pairings, as in [31].
- To handle the situation where the memory is not sufficient to store all the precomputed data for all vector elements, we also propose an adaptive method that can adjust the number of precomputed elements. Therefore, the performance of the proposed method can be fine-tuned adaptively by selectively storing the precomputed elements according to the storage capacity.
- We demonstrate that the proposed method can significantly accelerate the FE-based biometric verification process. In particular, we exploit the fact that the values related to a biometric template can be precomputed when registered as the template is stored once and repeatedly used without any change.

2. Preliminaries

2.1. Remote Biometric Verification System

In this study, we focus on biometric verification among the two types of biometric recognition. A biometric characteristic examiner can use a remote biometric verification system to authenticate users. Figure 1 shows the generic structure of the remote biometric verification system [1]. For better understanding, we will explain the structure with a facial verification system as an example. The system consists of a client for the user and a remote server for the biometric characteristic examiner. We can consider a smartphone as a typical example of the client. The following are the steps required to register a user's biometric data.

1. A user presents a biometric characteristic to a biosensor in a biometric capture subsystem, and the subsystem runs its biometric capture process to acquire a biometric sample. For facial verification, the user's smartphone camera takes a picture of the user's face. In addition to the biometric capture process, a biometric acquisition process is performed when required. The biometric acquisition process includes segmentation, quality control, and other preprocessing steps. For facial verification, various image processing techniques, such as face detection, alignment and frontalization, can be applied.
2. A client passes the biometric sample to the feature extraction module. The module attempts to extract a biometric feature from the biometric sample. For face verification, deep neural networks are frequently used [53,54].
3. The client performs biometric enrolment. To be precise, the client sends the extracted biometric feature to the server, and the server stores the biometric feature as a biometric reference to a biometric enrolment database. In the database, biometric references are managed using a data structure referred to as a biometric template.

In the authentication phase, the client performs the biometric capture and feature extraction processes, which are the same as the first and second steps of the above biometric registration phase. Then, the client sends the extracted biometric feature as the biometric claim to be used for verification. The server delivers the biometric feature from the client to its verifying module. This claimed feature is referred to as a biometric probe. The module loads the biometric reference (i.e., the stored biometric template) of the claimant for biometric comparison. Biometric comparison is the similarity estimation between the biometric reference and the biometric probe. For example, if the biometric feature is represented as a vector, the similarity can be estimated by computing either the Hamming or Euclidean distance between the two feature vectors. Based on the result of biometric comparison (i.e., the comparison score) and a decision policy including a threshold, whether the biometric probe and biometric reference have the same biometric source is determined. If this is affirmative, the server accepts the claimant as an authenticated user.

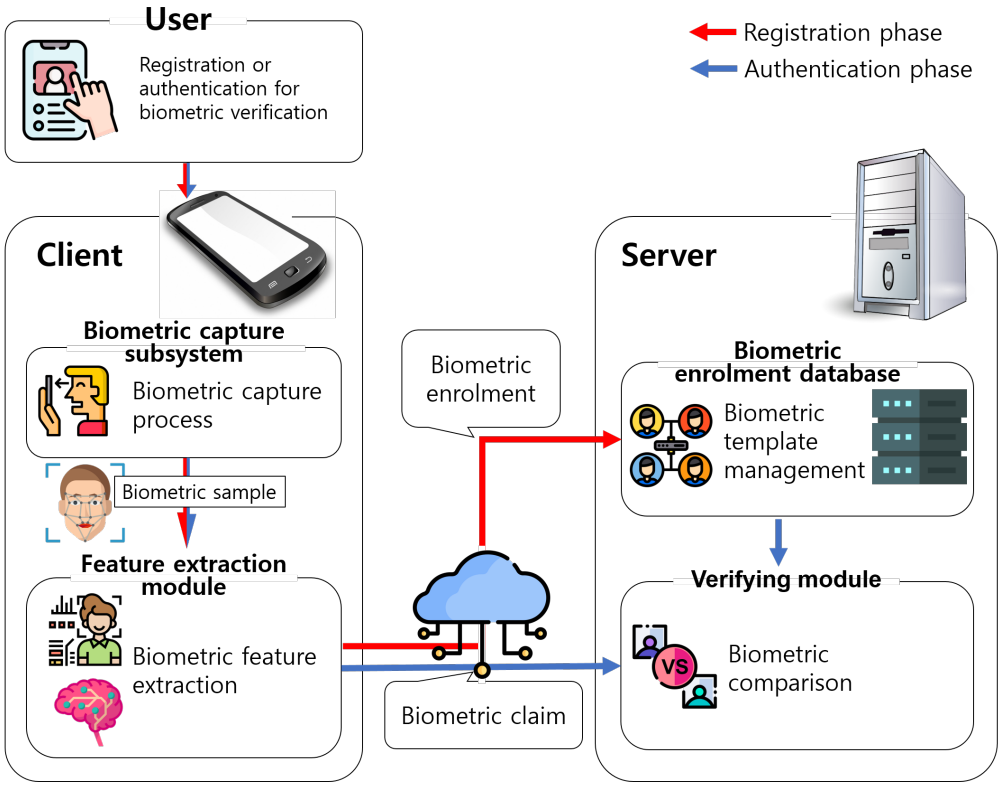

Figure 1. Generic structure of a remote biometric verification system.

However, as explained in the introduction, the above approach can be a threat to user's privacy, because the server may try to recover the user's biometric features from the stored template. Section 2.4 will briefly explain the countermeasure against this threat using functional encryption.

2.2. Barreto—Naehrig Curve (BN Curve)

If a non-supersingular elliptic curve over $\mathbb{F}_p$ contains a subgroup whose embedding degree k is not substantially large, it is called a pairing-friendly curve. In other words, computations in the field $\mathbb{F}_{p^k}$ are feasible. Barreto and Naehrig presented a method to construct pairing-friendly elliptic curves of prime order and embedding degree $k = 12$ [55], whose curve form is $E(\mathbb{F}_p) : y^2 = x^3 + b$, with $b \neq 0$. For non-zero t, they parameterized the order n of the elliptic curve group and the characteristic p as follows:

$$n = 36t^4 + 36t^3 + 18t^2 + 6t + 1 \quad (1)$$

$$p = 36t^4 + 36t^3 + 24t^2 + 6t + 1 \quad (2)$$

The order of a point $P \in E$ is the least non-zero integer r such that $[r]P = \infty$, where $[r]P$ is the sum of r terms equal to P [56]. Therefore, the fact that n is a prime implicitly indicates that r is equal to n, and r is also prime.

2.3. Pairing

2.3.1. Cryptographic Pairing

Pairing is defined as a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ for additive groups $\mathbb{G}_1$, $\mathbb{G}_2$ and a multiplicative group $\mathbb{G}_T$ [57]. The orders of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are the prime number r. The identity elements of these groups are denoted by $0_{\mathbb{G}_1}$, $0_{\mathbb{G}_2}$, and $1_{\mathbb{G}_T}$, respectively. Furthermore, the pairing should have the following two properties:

- Bilinearity For all $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_r$,

$$e([a]g_1, [b]g_2) = e(g_1, g_2)^{ab}. \tag{3}$$

- Non-degeneracy For $g_1 \neq 0_{\mathbb{G}_1}$, $g_2 \neq 0_{\mathbb{G}_2}$

$$e(g_1, g_2) \neq 1_{\mathbb{G}_T}. \tag{4}$$

In addition to the above two mathematical properties, cryptographic pairing requires the following property [57]:

- Computability The map e can be efficiently computed.

The most efficient cryptographic pairings are constructed using an elliptic curve E, which is defined over a finite field $\mathbb{F}_q$. Specifically, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of the rational points of an elliptic curve E defined over an extension $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$. Furthermore, $\mathbb{G}_T$ is the group $(\mathbb{F}^*_{q^k}, \times)$, where the group law is given by the field multiplication on $\mathbb{F}_{q^k}$. We define the tuple $(r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ as a *bilinear environment* of cryptographic pairing.

An inner-pairing product e_{prod} is defined by the following equation for two vectors $\mathbf{P} = (P_1, P_2, \ldots, P_d) \in \mathbb{G}_1^d$ and $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_d) \in \mathbb{G}_2^d$:

$$e_{prod}(\mathbf{P}, \mathbf{Q}) = \prod_{j=1}^{d} e(P_j, Q_j) \tag{5}$$

2.3.2. Miller's Algorithm and Final Exponentiation

The Weil pairing was first introduced by André Weil in 1940 [58]. It plays an important role in the theoretical study of the arithmetic of elliptic curves and Abelian varieties [58]. Miller presented an algorithm in 2004 that efficiently computes Weil pairing, which is the first practical pairing computation method [59]. Since then, most pairings, including Weil Pairing, have been designed based on Miller's algorithm for efficient operation. The basic Miller's algorithm takes a pair of elements of the elliptic curve subgroups $\mathbb{G}_1$ and $\mathbb{G}_2$, both of whose orders are the prime order r, and repeats a series of processes as much as the bit length m of r. This loop is referred to a *Miller loop*. For any point U, V on the elliptic curve and the element $X \in \mathbb{G}_1$, $Y \in \mathbb{G}_2$, line equation $L_{U,V}(X, Y)$ is defined as the equation of the line passing through U and V, whereas tangent equation $T_U(X, Y)$ is defined as the tangent to the point U. Miller's algorithm includes multiplication and squaring operations on $\mathbb{G}_T$, addition and multiplication operations on $\mathbb{G}_1$ or $\mathbb{G}_2$, and evaluation of the line and tangent equations. Miller's algorithm is used to compute not only Weil pairing but also many other cryptographic pairings, such as the Tate pairing or Tate variant pairings [60,61]. Thus, a special operation referred to as *final exponentiation* is performed to force the result of the Miller loop to be a unique value for the multiplicative group $\mathbb{G}_T$ [57]. In other words, final exponentiation must be performed after the Miller loop to obtain the correct operation result.

2.3.3. Optimal Ate Pairing on the BN Curve

The Ate pairing [62,63] and its variations [64–66] are simply optimized versions of the Tate pairing using Frobenius endomorphism [67]. In 2008, Vercauteren proposed optimal pairings and optimized the Miller loop of Ate pairing, which uses Frobenius endomorphism on a pairing-friendly elliptic curve [67]. In 2010, Beuchat et al. presented an implementation for the optimal Ate pairing of [67] on the BN curve [68]. They reported that the performance of Ate pairing is optimized by setting t of the BN curve as $2^{62} - 2^{54} + 2^{44}$. Algorithm 1 represents the algorithm for calculating the optimal Ate pairing [68], where $\pi(Q)$ is the Frobenius map of Q and $\pi^2(Q) = (\pi \cdot \pi)(Q)$.

Algorithm 1 Optimal Ate pairing on the BN curve.

Input: $s = 6t + 2$, $m =$ the bit length of s, $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$
Output: $e(P, Q)$

Write s in signed binary form, $s = \sum_{i=0}^{m-1} s[i]2^i$ with $s[i] \in \{-1, 0, 1\}$
$T \leftarrow Q, f \leftarrow 1$
for $i \leftarrow m - 2$ **down to** 0 **do**
    $f \leftarrow f^2 \cdot L_{T,T}(P), T \leftarrow 2T$
    if $s[i] = 1$ **then**
        $f \leftarrow f \cdot L_{T,Q}(P), T \leftarrow T + Q$
    else if $s[i] = -1$ **then**
        $f \leftarrow f \cdot L_{T,-Q}(P), T \leftarrow T - Q$
    end if
end for
$R \leftarrow \pi(Q), f \leftarrow f \cdot L_{T,R}(P), T \leftarrow T + R$
$R \leftarrow \pi^2(Q), f \leftarrow f \cdot L_{T,-R}(P), T \leftarrow T - R$
$f \leftarrow f^{(p^{12}-1)/r}$
return f

2.4. Function-Hiding Inner Product Encryption

Inner product encryption (IPE) is an FE whose function f is the inner product of the input vector $\mathbf{x}$ with the vector $\mathbf{y}$ encoded in the function f. That is, IPE performs $f(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$, by using a secret key associated with vector $\mathbf{y}$ and the ciphertext associated with vector $\mathbf{x}$ as inputs.

Meanwhile, if the FE guarantees that the data associated with its function f remain hidden as well as $\mathbf{x}$, we confirm that an FE has a function-hiding property. For example, the associated data may be vector $\mathbf{y}$ for IPE, and a privacy-preserving biometric verification system using IPE should be equipped with the function-hiding property as the biometric data should be securely handled both in the registration and authentication phases [69].

Hereinafter, we use $\prod_{IPE}$, a function-hiding IPE with practical performance proposed by Kim et al. [22]. For $\lambda \in \mathbb{N}$, $d \in \mathbb{N}$, and the range of the inner product S, the function-hiding IPE is defined as $\prod_{IPE} = (IPE.Setup, IPE.KeyGen, IPE.Encrypt, IPE.Decrypt)$, where each operation is defined as follows:

- $IPE.Setup(1^\lambda, S)$
    1. Select a bilinear environment $(r, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ according to the security parameter λ.
    2. Choose a matrix $\mathbf{B} \leftarrow \mathbb{GL}_d(\mathbb{Z}_r)$, where $\mathbb{GL}_d(\mathbb{Z}_r)$ refers to a group of $d \times d$ square matrix, where each element belongs to the finite field $\mathbb{Z}_r$ and an inverse matrix exists.
    3. Compute $\mathbf{B}^\star \leftarrow \det(\mathbf{B}) \cdot (\mathbf{B}^{-1})^\top$.
    4. Output the public parameter $pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, r, e, S)$ and the master secret key $msk = (pp, g_1, g_2, \mathbf{B}, \mathbf{B}^\star)$.
- $IPE.KeyGen(msk, \mathbf{y})$
    1. Choose a uniformly random element $\alpha \xleftarrow{R} \mathbb{Z}_r$.
    2. Using the master key msk and the vector $\mathbf{y} \in \mathbb{Z}_r^d$, output the secret key $sk = (K_1, K_2) = ([\alpha \cdot \det(\mathbf{B})]g_1, [\alpha \cdot \mathbf{y} \cdot \mathbf{B}]g_1)$, s.t. $K_2 \in \mathbb{G}_1^d$.
- $IPE.Encrypt(msk, \mathbf{x})$
    1. Choose a uniformly random element $\beta \xleftarrow{R} \mathbb{Z}_r$
    2. Using the master key msk and the vector $\mathbf{x} \in \mathbb{Z}_r^d$, output the ciphertext $ct = (C_1, C_2) = ([\beta]g_2, [\beta \cdot \mathbf{x} \cdot \mathbf{B}^\star]g_2)$, s.t. $C_2 \in \mathbb{G}_2^d$.
- $IPE.Decrypt(pp, sk, ct)$

1. Using the public parameter pp, the secret $sk = (K_1, K_2)$, and the ciphertext $ct = (C_1, C_2)$, compute $D_1 = e(K_1, C_1)$ and $D_2 = e_{prod}(K_2, C_2)$.
2. Find a solution z for $D_1^z = D_2$ If this z exists, it satisfies $z = \langle \mathbf{x}, \mathbf{y} \rangle$. Output z if it exists; otherwise, output $\perp$, indicating that a solution does not exist.

In this study, we use $\prod_{IPE}$ to construct a privacy-preserving biometric verification system. The two vectors $\mathbf{x}$ and $\mathbf{y}$ are encoded to ensure that they represent the biometric probe and biometric reference, respectively. Furthermore, the authors of [22] suggested methods to encode a biometric feature vector to either $\mathbf{x}$ or $\mathbf{y}$; thus, the Hamming and the Euclidean distance between two biometric feature vectors can be calculated using the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$. Using this method, we can encrypt all biometric data transmitted to the server as well as the stored biometric template. The biometric comparison is performed on the encrypted biometric data. However, the comparison score is output as a plain value to ensure that the verification module can decide regarding the authenticity of the claimant. In summary, a biometric verification system that keeps all biometric data from leaking the biometric characteristics even during the biometric comparison can be constructed using $\prod_{IPE}$.

3. Existing Optimization Techniques for Computing Pairing

The inner-pairing product can be performed in a naive manner to calculate $e(P_j, Q_j)$ for all j and multiply them all. We call this approach the *Naive method*. However, this native method can be improved based on two directions of research. We briefly review them in this section.

3.1. Optimal Ate Pairing Product on the BN Curve

In 2005, Scott proposed three ideas to optimize the Naive method for Tate pairing [31].

1. In the case of a modular multiplicative inverse operation, a simultaneous inversion operation [70] can be applied.
2. During the computation of Miller's algorithm, the squaring operation on $\mathbb{G}_T$ (e.g., in line 4 of Algorithm 1) can be shared.
3. The final exponentiation operation can be shared.

In 2006, Robert Granger et al. reported that the performance of inner-pairing product computation for Ate pairings can be improved by applying the second and third ideas [32]. In 2015, Eric Zavattoni et al. implemented an optimized method to compute the products of optimal Ate pairings on the BN curve, which we call the *Product method* in this paper [33].

Algorithm 2 demonstrates how the Product method is computed in the bilinear environment of Section 2.3.3.

3.2. Fixed Argument Pairings

In 2010, Costello et al. proposed a method to compute a pairing using precomputation when one argument of the pairing is fixed [34]. Algorithm 3 demonstrates the application of precomputation to Q for the optimal Ate pairing, adopting the method in [34] when $Q \in \mathbb{G}_2$ is fixed. Algorithm 4 presents the pairing computation procedure when P and Q' are given as inputs, where Q' is a precomputed tuple based on Q. The main idea of this precomputation-based method is that the line and the tangent equation in the Miller loop of the optimal Ate pairing can be precomputed with only Q without P to obtain the gradient λ and constant c of the equations. π can also be computed in advance with only Q. Thus, it is also included in precomputation.

During online computation of the pairing, the precomputed Q', rather than Q, is applied to the linear equation, tangent equation, and π. The *Fixed-Q method* refers to an inner-pairing product method that replaces the individual pairing of the Naive method to the online computation of Fixed-Q pairing.

Algorithm 2 Products of optimal Ate pairings on the BN curve (Product method).

Input: $s = 6t + 2$, $m =$ the bit length of s, $P_j \in \mathbb{G}_1$, $Q_j \in \mathbb{G}_2$, where j is $1, ..., d$
Output: $e_{prod}(\mathbf{P}, \mathbf{Q})$

Write s in signed binary form, $s = \sum_{i=0}^{m-1} s[i]2^i$ with $s[i] \in \{-1, 0, 1\}$
$f \leftarrow 1$
for $j \leftarrow 1$ **to** n **do**
    $T_j \leftarrow Q_j$
end for
for $i \leftarrow m - 2$ **down to** 0 **do**
    $f \leftarrow f^2$
    for $j \leftarrow 1$ **to** d **do**
        $f \leftarrow f \cdot L_{T_j,T_j}(P_j), T_j \leftarrow [2]T_j$
        if $s[i] = 1$ **then**
            $f \leftarrow f \cdot L_{T_j,Q_j}(P_j), T_j \leftarrow T_j + Q_j$
        else if $s[i] = -1$ **then**
            $f \leftarrow f \cdot L_{T_j,-Q_j}(P_j), T_j \leftarrow T_j - Q_j$
        end if
    end for
end for
for $j \leftarrow 1$ **to** d **do**
    $R \leftarrow \pi(Q_j), f \leftarrow f \cdot L_{T_j,R}(P_j), T_j \leftarrow T_j + R$
    $R \leftarrow \pi^2(Q_j), f \leftarrow f \cdot L_{T_j,-R}(P_j), T_j \leftarrow T_j - R$
end for
$f \leftarrow f^{(p^{12}-1)/r}$
return f

Algorithm 3 Fixed-Q precomputation.

Input: $s = 6t + 2$, $m =$ the bit length of s, $Q \in \mathbb{G}_2$
Output: $Q' = (G_{DBL}, G_{ADD}, \pi_Q, {\pi^2}_Q)$

Write s in signed binary form, $s = \sum_{i=0}^{m-1} s[i]2^i$ with $s[i] \in \{-1, 0, 1\}$
$T \leftarrow Q, G_{DBL} \leftarrow \{\emptyset\}, G_{ADD} \leftarrow \{\emptyset\}$
for $i \leftarrow m - 2$ **down to** 0 **do**
    Compute λ and c, such that $y_Q + \lambda x_Q + c$ is the line tangent to T
    $T \leftarrow [2]T$
    Append (λ, c) to G_{DBL}
    if $s[i] = 1$ **then**
        Compute λ and c, such that $y_Q + \lambda x_Q + c$ is the line joining T and Q
        $T \leftarrow T + Q$
        Append (λ, c) to G_{ADD}
    else if $s[i] = -1$ **then**
        Compute λ and c, such that $y_{-Q} + \lambda x_{-Q} + c$ is the line joining T and $-Q$
        $T \leftarrow T - Q$
        Append (λ, c) to G_{ADD}
    end if
end for
$R \leftarrow \pi(Q)$
Compute λ and c, such that $y_R + \lambda x_R + c$ is the line joining T and R
$\pi_Q \leftarrow (\lambda, c)$
$R \leftarrow -\pi^2(Q)$
Compute λ and c, such that $y_R + \lambda x_R + c$ is the line joining T and R
${\pi^2}_Q \leftarrow (\lambda, c)$
$Q' \leftarrow (G_{DBL}, G_{ADD}, \pi_Q, {\pi^2}_Q)$
return Q'

Algorithm 4 Fixed-Q pairing.

Input: $s = 6t + 2$, m = the bit length of s, $P \in \mathbb{G}_1$, $Q' = (G_{DBL}, G_{ADD}, \pi_Q, \pi^2{}_Q)$ (the precomputation tuple for Q, where $Q \in \mathbb{G}_2$)
Output: $e(P, Q)$

1: Write s in signed binary form, $s = \sum_{i=0}^{m-1} s[i]2^i$ with $s[i] \in \{-1, 0, 1\}$
2: $f \leftarrow 1, cnt \leftarrow 1$
3: **for** $i \leftarrow m - 2$ **down to** 0 **do**
4: $\quad \lambda, c \leftarrow G_{DBL}[i]$
5: $\quad$ Compute $g \leftarrow (y_P + \lambda x_P + c)$
6: $\quad f \leftarrow f^2 \cdot g$
7: $\quad$ **if** $s[i] \neq 0$ **then**
8: $\quad\quad \lambda, c \leftarrow G_{ADD}[cnt]$
9: $\quad\quad$ Compute $g \leftarrow (y_P + \lambda x_P + c)$
10: $\quad\quad cnt \leftarrow cnt + 1$
11: $\quad\quad f \leftarrow f \cdot g$
12: $\quad$ **end if**
13: **end for**
14: $\lambda, c \leftarrow \pi_Q$
15: Compute $g \leftarrow (y_P + \lambda x_P + c)$
16: $f \leftarrow f \cdot g$
17: $\lambda, c \leftarrow \pi^2{}_Q$
18: Compute $g \leftarrow (y_P + \lambda x_P + c)$
19: $f \leftarrow f \cdot g$
20: $f \leftarrow f^{(p^{12}-1)/r}$
21: **return** f

4. Proposed Method

In this section, we present our proposed method to efficiently compute an inner-pairing product. The proposed method combines two previous approaches and adopts both precomputation and shared computation techniques. We call our method the *Fixed-Q Product* method. This method is a revised version of the method presented in the preliminary version of this paper [71] and the Master's Thesis of the first author [72].

Algorithm 5 presents the detailed procedure of the Fixed-Q Product method. In the input of the Fixed-Q Product method, $\mathbf{Q}'$ is used instead of $\mathbf{Q}$, unlike the Product method. In other words, all elements of $\mathbf{Q}$ in Algorithm 2 are now used to obtain $\mathbf{Q}'$ using the Fixed-Q precomputation (Algorithm 3) in advance.

We initialize a few variables in lines 1–5 of Algorithm 5 prior to performing the Miller loop. The parameter s is expanded as a signed binary form. The variable f for accumulating the products of the pairings is set to 1, and all elements of the array cnt for G_{ADD} are initialized to 1.

After initialization, the Miller loop runs in lines 6–20 of Algorithm 5. Unlike in a single pairing, the Miller loop has the form of a nested loop as it computes multiple pairings. In the case of the Naive method, the inside loop should have been executed many times, depending on the number of inputs d. The nested loop structure of Algorithm 5 is the same as that of the Product method to share the squaring operation. In other words, the code of lines 7–19 is repeated by the length of s by using i. In each iteration of this outer loop, lines 9–18 are repeated based on the number of inputs d using j in the inside loop. Through application of this nested loop, the proposed method can improve the performance of the Fixed-Q method similar to how the Product method improves the Naive method. Furthermore, we can describe the effect of our method in the aspect of the amount of online computation. In other words, unlike the Product method, the proposed method performs the Fixed-Q pairing using Q'_j in the code of lines 9–18. To support this improvement, each array element $cnt[j]$ plays the role of the variable cnt of Algorithm 4.

Algorithm 5 Fixed-Q Product method.

Input: $s = 6t + 2$, m = the bit length of s, $\mathbf{P} = \{(P_1, ..., P_d) \mid P_j \in \mathbb{G}_1\}$, $\mathbf{Q}' = \{(Q'_1, ..., Q'_d) \mid Q'_j$ is the precomputation tuple for $Q_j \in \mathbb{G}_2\}$

Output: $e_{prod}(\mathbf{P}, \mathbf{Q})$

1: Write s in signed binary form, $s = \sum_{i=0}^{m-1} s[i]2^i$ with $s[i] \in \{-1, 0, 1\}$
2: $f \leftarrow 1$
3: **for** $j \leftarrow 1$ **to** d **do**
4: $\quad cnt[j] \leftarrow 1$
5: **end for**
6: **for** $i \leftarrow m - 2$ **down to** 0 **do**
7: $\quad f \leftarrow f^2$
8: $\quad$ **for** $j \leftarrow 1$ **to** d **do**
9: $\quad\quad G_{DBL}, G_{ADD} \leftarrow Q'_j$
10: $\quad\quad \lambda, c \leftarrow G_{DBL}[i]$
11: $\quad\quad$ Compute $g \leftarrow (y_P + \lambda x_P + c)$
12: $\quad\quad f \leftarrow f \cdot g$
13: $\quad\quad$ **if** $s[i] \neq 0$ **then**
14: $\quad\quad\quad \lambda, c \leftarrow G_{ADD}[cnt[j]]$
15: $\quad\quad\quad$ Compute $g \leftarrow (y_P + \lambda x_P + c)$
16: $\quad\quad\quad cnt[j] \leftarrow cnt[j] + 1$
17: $\quad\quad\quad f \leftarrow f \cdot g$
18: $\quad\quad$ **end if**
19: $\quad$ **end for**
20: **end for**
21: **for** $j \leftarrow 1$ **to** d **do**
22: $\quad \pi'_Q, (\pi')^2_Q \leftarrow Q'_j$
23: $\quad \lambda, c \leftarrow \pi'_Q$
24: $\quad$ Compute $g \leftarrow (y_P + \lambda x_P + c)$
25: $\quad f \leftarrow f \cdot g$
26: $\quad \lambda, c \leftarrow (\pi')^2_Q$
27: $\quad$ Compute $g \leftarrow (y_P + \lambda x_P + c)$
28: $\quad f \leftarrow f \cdot g$
29: **end for**
30: $f \leftarrow f^{(p^{12}-1)/r}$
31: **return** f

The Frobenius map and final exponentiation should be applied to the optimal Ate pairing after the Miller loop. As the operation of the Frobenius map is only related to Q, all the Frobenius maps of individual pairings can be included in the Fixed-Q precomputation. Therefore, we apply the precomputed Frobenius maps to the proposed method by using π'_Q and $(\pi')^2_Q$ from Q'_j in lines 21–29. As mentioned in Section 3.1, the final exponentiation is a shareable operation. Thus, the final exponentiation can be performed only once after the Miller loop (in line 30).

It should be noted that the speedup of the proposed method is obtained at the expense of additional memory to store $\mathbf{Q}'$. The exact amount of memory required to store the d tuples $Q'_1, ..., Q'_d$ of $\mathbf{Q}'$ will be analyzed in the next section. To handle the situation where the memory budget is very tight, we propose an adaptive method that adjusts the number of precomputed tuples according to the storage capacity. Algorithm 6 is this modified version for the situation where only the memory for $k(\leq d)$ precomputed tuples is available. Without loss of generality, we assume that only the precomputed tuples $Q'_1, ..., Q'_k$ are given. Therefore, Algorithm 6 takes as input these tuples as well as $Q_{k+1}, \ldots, Q_d$, the field elements for the non-precomputed portion. Algorithm 6 can be viewed as the combination of Algorithm 5 and Algorithm 2. Its main loop is the same as that of Algorithm 5. However, it has lines 6–8, lines 23–30, and lines 41–44, i.e., the routines to handle the operations

corresponding to non-precomputed elements. By adjusting the parameter k, the algorithm can be adapted to the current memory capacity. That is, Algorithm 6 has a time-memory trade-off. The relation between the number of precomputed tuples and the speed will be precisely analyzed in the next section.

Algorithm 6 Adaptive method.

Input: $s = 6t + 2$, m = the bit length of s, $\mathbf{P} = \{(P_1, ..., P_d) \mid P_j \in \mathbb{G}_1\}$, $\mathbf{Q''} = \{(Q'_1, ..., Q'_k, Q_{k+1}, ..., Q_d) \mid Q'_j$ is the precomputation tuple for $Q_j \in \mathbb{G}_2, 1 < k \leq d\}$

Output: $e_{prod}(\mathbf{P}, \mathbf{Q})$

Write s in signed binary form, $s = \sum_{i=0}^{m-1} s[i]2^i$ with $s[i] \in \{-1, 0, 1\}$
$f \leftarrow 1$
for $j \leftarrow 1$ **to** k **do**
    $cnt[j] \leftarrow 1$
end for
for $j \leftarrow k + 1$ **to** d **do**
    $T_j \leftarrow Q_j$
end for
for $i \leftarrow m - 2$ **down to** 0 **do**
    $f \leftarrow f^2$
    for $j \leftarrow 1$ **to** k **do**
        $G_{DBL}, G_{ADD} \leftarrow Q'_j$
        $\lambda, c \leftarrow G_{DBL}[i]$
        Compute $g \leftarrow (y_P + \lambda x_P + c)$
        $f \leftarrow f \cdot g$
        if $s[i] \neq 0$ **then**
            $\lambda, c \leftarrow G_{ADD}[cnt[j]]$
            Compute $g \leftarrow (y_P + \lambda x_P + c)$
            $cnt[j] \leftarrow cnt[j] + 1$
            $f \leftarrow f \cdot g$
        end if
    end for
    for $j \leftarrow k + 1$ **to** d **do**
        $f \leftarrow f \cdot L_{T_j, T_j}(P_j), T_j \leftarrow [2]T_j$
        if $s[i] = 1$ **then**
            $f \leftarrow f \cdot L_{T_j, Q_j}(P_j), T_j \leftarrow T_j + Q_j$
        else if $s[i] = -1$ **then**
            $f \leftarrow f \cdot L_{T_j, -Q_j}(P_j), T_j \leftarrow T_j - Q_j$
        end if
    end for
end for
for $j \leftarrow 1$ **to** k **do**
    $\pi'_Q, (\pi')^2_Q \leftarrow Q'_j$
    $\lambda, c \leftarrow \pi'_Q$
    Compute $g \leftarrow (y_P + \lambda x_P + c)$
    $f \leftarrow f \cdot g$
    $\lambda, c \leftarrow (\pi')^2_Q$
    Compute $g \leftarrow (y_P + \lambda x_P + c)$
    $f \leftarrow f \cdot g$
end for
for $j \leftarrow k + 1$ **to** d **do**
    $R \leftarrow \pi(Q_j), f \leftarrow f \cdot L_{T_j, R}(P_j), T_j \leftarrow T_j + R$
    $R \leftarrow \pi^2(Q_j), f \leftarrow f \cdot L_{T_j, -R}(P_j), T_j \leftarrow T_j - R$
end for
$f \leftarrow f^{(p^{12}-1)/r}$
return f

5. Performance Analysis

5.1. Theoretical Analysis

In this subsection, we analyze the expected amount of computation and storage required for the Naive, Product, Fixed-Q, and the proposed Fixed-Q Product methods. First, we denote the total amount of computation required for the basic optimal Ate pairing algorithm (Algorithm 1) and the Fixed-Q pairing algorithm (Algorithm 4) as C_{basic} and C_{fixed}, respectively. As explained in Section 3.2, C_{fixed} is significantly smaller than C_{basic}. We also denote the amount of computation required for a squaring operation on $\mathbb{G}_T$ and the final exponentiation as C_{sqr} and C_{fin}, respectively. Subsequently, the amount of computation required to compute an inner-pairing product (5) of two d-dimensional vectors can be expressed as follows:

- Naive method
$$C_{basic} \times d \tag{6}$$
- Product method
$$(C_{basic} - C_{sqr} - C_{fin}) \times d + C_{sqr} + C_{fin} \tag{7}$$
- Fixed-Q method
$$C_{fixed} \times d \tag{8}$$
- Fixed-Q Product method (proposed)
$$(C_{fixed} - C_{sqr} - C_{fin}) \times d + C_{sqr} + C_{fin} \tag{9}$$

Noticeably, the computational costs of all methods are represented as linear functions in d. However, comparing (6) and (7), we can observe that the term $C_{sqr} + C_{fin}$ was moved from the slope to the constant intercept part, thus, significantly reducing the slope. The same relation holds between (8) and (9). In particular, the cost reduction of the proposed method over the Fixed-Q method (and that of the Product method over the Naive method) is $(d-1)(C_{sqr} + C_{fin})$. Consequently, the proposed method is expected to be the fastest among the four methods when $d \geq 2$.

The amount of computation for the Fixed-Q method and the proposed method is reduced at the expense of memory to store the precomputed elements. In other words, a time-memory trade-off occurs. To estimate the additional memory required for storing the precomputed elements, we expressed the bit length of the data structure Q' that represents the precomputation table. Q' is a tuple composed of $(G_{DBL}, G_{ADD}, \pi_Q, {\pi^2}_Q)$. According to Algorithm 3, G_{DBL} and G_{ADD} are the arrays with their elements in $\mathbb{F}_p \times \mathbb{F}_p$. Whenever the code of line 6 in Algorithm 3 is run, the number of elements in the G_{DBL} array is, thus, increased by one, and this line is repeated by $m-1$ times. Thereafter, G_{DBL} will contain $m-1$ elements in $\mathbb{F}_p \times \mathbb{F}_p$. The codes of line 10 and 14 also increase the length of G_{ADD} by one. However, these lines are executed only when the corresponding element $s[i]$ in the signed binary representation of s is non-zero. For given $t = 2^{62} - 2^{52} + 2^{44}$, the number of non-zero terms in the signed binary representation of s is exactly seven. Therefore, G_{ADD} will finally contain seven elements in $\mathbb{F}_p \times \mathbb{F}_p$. Finally, the codes of lines 19 and 22 are executed once for π_Q and ${\pi^2}_Q$. Therefore, $\pi_Q, {\pi^2}_Q \in \mathbb{F}_p \times \mathbb{F}_p$. As m, the bit length of $s = 6t + 2$, is 65, we can observe that $Q' \in (\mathbb{F}_p \times \mathbb{F}_p)^{(m-1)+7+2} = (\mathbb{F}_p \times \mathbb{F}_p)^{73}$, and the bit length of Q' can be expressed as $73 \times 2 \times l = 146l$, where l denotes the bit length of prime p. In other words, $146l$ bits are additionally required to store the precomputed data Q' in Algorithm 3 and use it in Algorithm 4. For Algorithm 5, we need d tables $Q'_1, \ldots, Q'_d$, requiring $146dl$ bits of the precomputation storage.

5.2. Experimental Results

To verify the performance improvement, we implemented the proposed method as well as the three previous methods, and then we compared their performance in terms of execution time and memory usage. This experiment was conducted on a desktop PC with

an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz, 16GB memory, and Ubuntu Desktop 16.04 LTS. The program was written in C++. In particular, GMP 6.1.2 [73], MCL 1.05 [74], and NTL 11.3.2 [75] libraries were applied for algebraic operations.

According to the theoretical analysis in Section 5.1, the amount of computation for each method can be expressed as a linear equation in the number d of the input pairs. The slope of the linear equation and the value of the y-intercept can be estimated by measuring the execution times of the component operations. Therefore, we measured the durations required for a squaring operation and final exponentiation and as those for pairing operations. Table 1 presents the result of this measurement. The figures in Table 1 are the average of the execution times in 10,000 executions with a random input for each operation, and the unit of execution time is 10^6 CPU clocks (Mclk).

Table 1. Execution times of the operations constituting an inner-pairing product.

Operation Name	Execution Time (Mclk)
Squaring on $\mathbb{G}_T$	0.008
Final exponentiation	0.629
Basic optimal Ate pairing (Algorithm 1)	1.595
Fixed-Q pairing (Algorithm 4)	1.411

Using the measured data, we estimated the slopes and y-intercepts in the linear Equations (6)–(9) in the previous subsection. The cost of the squaring operation C_{sqr} can be estimated as $m\times$(the execution time for a single squaring operation), where m (i.e., the bit length of $s = 6t + 2$) is calculated as 65 given that $t = 2^{62} - 2^{54} + 2^{44}$ for the optimal Ate pairing. Consequently, we obtained the following expressions for the execution times of the four inner-pairing product methods, which are also summarized in Table 2:

- Naive method:
$$C_{Naive} = 1.595d \tag{10}$$
- Product method:
$$C_{Prod} = (1.595 - 0.008 \times 65 - 0.629)d + 0.008 \times 65 + 0.629 = 0.464d + 1.131 \tag{11}$$
- Fixed-Q method:
$$C_{FixedQ} = 1.411d \tag{12}$$
- Fixed-Q Product method (proposed):
$$C_{FixedQProd} = (1.411 - 0.008 \times 65 - 0.629)d + 0.008 \times 65 + 0.629 = 0.280d + 1.131 \tag{13}$$

Table 2. Execution times of the inner-pairing product computation for various methods.

Methods	Expected Cost	Measured Cost
Naive	$C_{basic} \times d$	$1.595d$
Product	$(C_{basic} - C_{sqr} - C_{fin}) \times d + C_{sqr} + C_{fin}$	$0.464d + 1.131$
Fixed-Q	$C_{fixed} \times d$	$1.411d$
Proposed	$(C_{fixed} - C_{sqr} - C_{fin}) \times d + C_{sqr} + C_{fin}$	$0.280d + 1.131$

We also verified the validity of the expressions given above by directly measuring the execution times of the inner-pairing product computation. Furthermore, we measured these execution times, increasing d from 10 to 1000 by 10. For each combination, we measured the execution time of each method 1000 times. Figure 2 presents the average execution times of the four inner product methods. Certainly, the execution times of the four methods increase almost linearly with d. Figure 3 presents the relative ratio of the execution time of each method to that of the Naive method. We can observe that the ratio remains as a constant for each method, except the region for small d, where the

influences of constant terms in (7) and (9) are non-negligible. Please note that the amounts of computation in the Naive and Fixed-Q methods (Equations (6) and (8)) are proportional to d, but those in the Product and Fixed-Q Product methods are not exactly proportional to d. However, when d gets sufficiently large, the constant terms in (7) and (9) almost do not affect the overall performance, and (7) and (9) become almost proportional to d. For this region with sufficiently large d, the relative ratio of the execution time is essentially the ratio of the slopes. For example, the ratio of the Fixed-Q Product method over the Naive method is approximately $(C_{fixed} - C_{sqr} - C_{fin})/(C_{basic})$ for large d, whereas it is originally $((C_{fixed} - C_{sqr} - C_{fin})d + (C_{sqr} + C_{fin}))/(C_{basic}d)$. This explains the slightly bent portions in the curves for the Product and Fixed-Q Product methods in Figure 3.

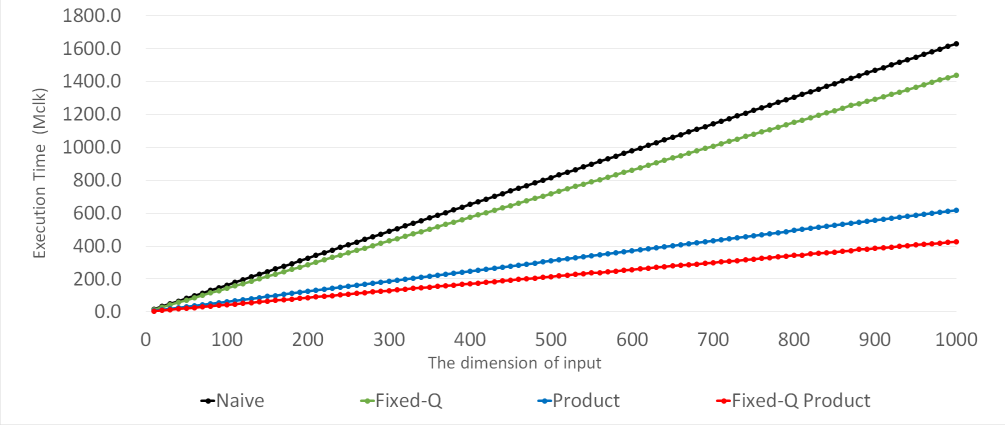

Figure 2. Performance comparison according to the dimension of input for each method.

In other words, four horizontal lines are almost parallel to the d-axis. The constant ratios are 0.881, 0.381, and 0.264 for the Fixed-Q, Product, and proposed methods, respectively. This implies that the proposed method improves the performance of the Naive, Fixed-Q, and Product methods by 3.8, 3.3, and 1.4 times, respectively. If we compare the proposed method with the best previous method, Product, we see that the proposed method reduces the execution time of the Product method by $1 - 0.264/0.381 \approx 30.7\%$.

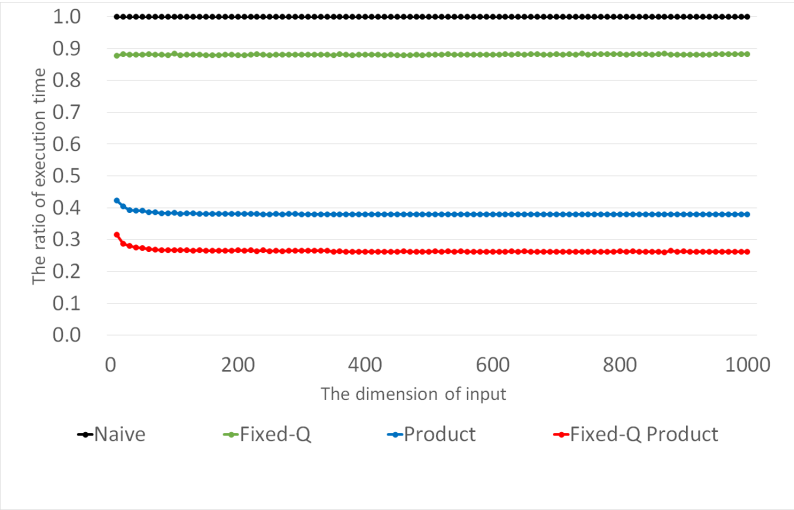

Figure 3. Comparison of the performance ratio of other methods to compute inner-pairing products compared to the Naive method.

The amount of storage space required for each method was also measured and analyzed. The software module used in this study uses a 256-bit data type to express an element in $\mathbb{F}_p$. Each element in $\mathbb{G}_1$ is a point on an elliptic curve defined over $\mathbb{F}_p$. Therefore, it is composed of two elements in $\mathbb{F}_p$ to represent x and y coordinates. Each element in $\mathbb{G}_2$

is a point on an elliptic curve defined over an extension field. Therefore, it is composed of four elements in $\mathbb{F}_p$. Subsequently, the sizes of the data structures to handle the elements in $\mathbb{G}_1$ and $\mathbb{G}_2$ are $512(= 2 \times 256)$ and $1024(= 4 \times 256)$ bits, respectively. The data structure for a precomputation table Q' consumes $146 \times 256 = 37,376$ bits. When an inner-pairing product is computed with two d-dimensional vectors using the proposed method, $37,376d$ bits are required for the precomputation table. For example, considering that $d = 130$, which is a typical value for our biometric verification application, this amount corresponds to only 0.6 MB. As explained in Section 4, only a subset of precomputed elements may be computed if the storage is not sufficient. Figure 4 demonstrates the time-memory trade-off of Algorithm 6 for various values of k when $d = 130$. The x-axis represents the amount of available memory. k varies from 0 to 130. When $k = 130$, approximately 0.6 MB is required. As the graph shows, the execution time of Algorithm 6 linearly decreases according to the increase in the amount of memory.

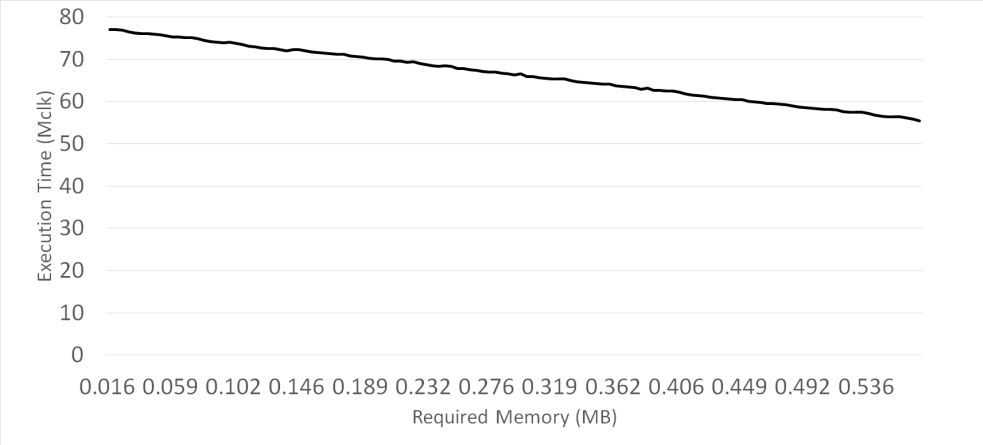

Figure 4. Time-memory trade-off of the adaptive method.

Finally, we verify whether the Fixed-Q and Fixed-Q Product methods consume additional communication bandwidth for the transmission of the precomputation tables. When the inner-pairing product operation is applied to privacy-preserving remote biometric verification, this operation is performed by the receiver (i.e., by the remote authentication server) to conduct $IPE.Decrypt$ with two input vectors $\mathbf{P} = (P_1, P_2, \ldots, P_d) \in \mathbb{G}_1^d$ and $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_d) \in \mathbb{G}_2^d$. The second vector, $\mathbf{Q}$, is related to the stored biometric template. Therefore, it is transmitted in the registration phase, and the required precomputation can be conducted on the server side. In other words, Algorithm 3 is performed for each element in $\mathbf{Q}$ by the server. Therefore, additional communication bandwidth is not required.

6. Application

As explained in Section 2.4, if FE is applied to a remote biometric verification system, the client may encrypt and securely transmit the user's biometric data, the server may store the encrypted biometric template, and the biometric comparison may be performed while all biometric data remain encrypted. In this section, we construct a simple facial verification system using $\prod_{IPE}$ and describe how the FE can be applied to a remote biometric verification system, following the idea presented in [22]. We also demonstrate that the proposed method significantly improves the facial verification performance. For biometric comparison, the server uses Euclidean distance as the metric for the similarity between two feature vectors, which is the most widely used approach in biometric authentication. Furthermore, a vector encoding method proposed by Kim [22] is adopted to perform this comparison in the ciphertext domain. When two d-dimensional feature vectors $\mathbf{x}$ and $\mathbf{y}$ are given as a biometric reference and biometric probe, respectively, the similarity score is defined as $\|\mathbf{x} - \mathbf{y}\|^2$ (i.e., the square of the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$). To compute $\|\mathbf{x} - \mathbf{y}\|^2$ using $\prod_{IPE}$, the following three operations are defined:

- $EncodeX(msk, \mathbf{x})$
  1. Construct a $(d+2)$-dimensional vector $\mathbf{x}' = (\|\mathbf{x}\|^2, -2x_1, -2x_2, \ldots, -2x_d, 1)$ from $\mathbf{x} = (x_1, \ldots, x_d)$.
  2. Output $ct = IPE.Encrypt(msk, \mathbf{x}')$.
- $EncodeY(msk, \mathbf{y})$
  1. Construct a $(d+2)$-dimensional vector $\mathbf{y}' = (1, y_1, y_2, \ldots, y_d, \|\mathbf{y}\|^2)$ from $\mathbf{y} = (y_1, \ldots, y_d)$.
  2. Output $sk = IPE.KeyGen(msk, \mathbf{y}')$.
- $Euclid(pp, sk, ct)$
  1. Calculate $z = IPE.Decrypt(pp, sk, ct)$.
  2. Output z. (z satisfies $z = \langle \mathbf{x}', \mathbf{y}' \rangle = (\|\mathbf{x}\|^2 - 2x_1y_1 - 2x_2y_2 - \cdots - 2x_dy_d + \|\mathbf{y}\|^2) = (\|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{y} \rangle + \|\mathbf{y}\|^2) = \|\mathbf{x} - \mathbf{y}\|^2$).

In our biometric verification system, *IPE.Encrypt* (i.e., *EncodeX*) will be used to protect the biometric template in the registration phase. Meanwhile, *IPE.KeyGen* (i.e., *EncodeY*) will be used to protect the biometric probe of the claimant in the authentication phase. We might have used the IPE functions in other ways (i.e., *IPE.KeyGen* for registration and *IPE.Encrypt* for authentication). However, we made the above choice owing to the following reason: the stored biometric template does not change frequently after registration, whereas the biometric probe for authentication changes every session. Thus, applying precomputation to the stored template is suitable. According to Algorithm 5, the precomputation is only applicable to the second argument of $e_{prod}(\mathbf{P}, \mathbf{Q})$, that is, $\mathbf{Q}$, whose elements are from $\mathbb{G}_2$. According to the description of $\prod_{IPE}$ in Section 2.4, this $\mathbf{Q}$ should be C_2, which is computed by *IPE.Encrypt*.

Figure 5 shows the simple facial verification system that uses the above three operations. *EncodeX* and *EncodeY* are performed by the client. Meanwhile, *Euclid* is performed by the server and includes an inner-pairing product for *IPE.Decrypt*. In other words, if the inner-pairing product can be accelerated, we can also improve the performance of *Euclid*.

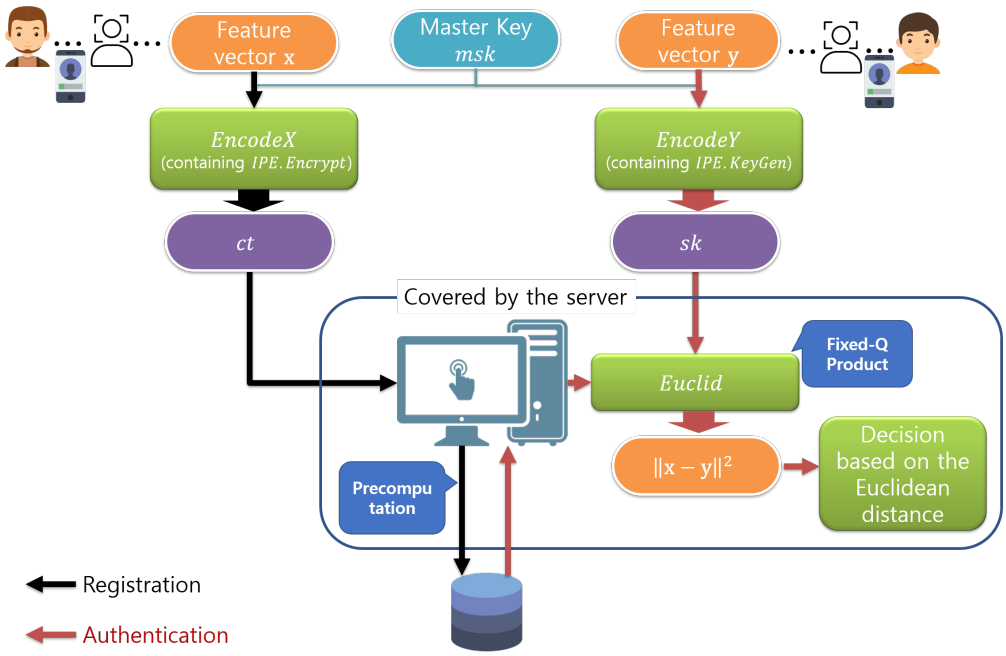

Figure 5. Simple facial verification system that uses $\prod_{IPE}$ with the proposed method.

As shown in Figure 5, the client transmits a user's encrypted feature vector as a biometric reference to the server, and then the server stores the encrypted vector as a biometric template in the registration phase. Using the proposed method, the server can

perform the Fixed-Q precomputation on all the elements of the biometric template and use the precomputed data to accelerate the inner-pairing product computation in *Euclid*.

To estimate the effect of the proposed method in the performance of a remote biometric verification system, we emulated a remote facial verification system where the client comprises an image processing module that processes the face image and produces a feature vector, and a cryptographic module for the *EncodeX* and *EncodeY* operations. The server performs the biometric comparison using a cryptographic module implementing the *Euclid* operation. We did not actually implement the image processing module, but emulated the one provided in [2]. We brought a part of the experimental results for image processing time in [2], and actually measured the durations for $\prod_{IPE}$ operations. Combining both data, we estimated the overall time for biometric verification. To measure the times for $\prod_{IPE}$ operations, we used the same mobile device and PC as used in [2]. Table 3 presents our experimental results. The face image processing column covers the entire image processing procedure, i.e., from biometric capture and acquisition process and feature extraction. In [2], four datasets were provided—Auto, Guide, CFP, and ORL. Each dataset uses 128-dimensional feature vectors in common. Therefore, we measured the performance of *EncodeY* with 128-dimensional feature vectors. However, we did not compare the performance of *EncodeX* as it is called only once for registration. In particular, we measured the performance of *Euclid* by dividing it into two parts, Pairing and DLP. Pairing is the part that consists of a single pairing operation for D_1 and e_{prod} for D_2 in $IPE.Decrypt$. For comparison, we measured the performance of the inner-pairing product, e_{prod}, using the proposed and Naive methods. DLP is the part that finds a solution to the discrete logarithm problem $D_1^z = D_2$. The total execution time for authentication comprises face image processing, *EncodeY*, and *Euclid*, and presents the data when the Naive method and the proposed method are applied. The last column provides the ratio of reduced time over the Naive method (i.e., $(1 - \frac{Total\ (proposed)}{Total\ (Naive)}) \times 100\%$). From Table 3, the proposed Fixed-Q Product method reduces the time for *Euclid* operation by $\left(1 - \frac{15.66+120.30}{58.86+120.30}\right) \times 100\% = 24.1\%$ compared to the Naive method, and the overall authentication time is also expected to be reduced by 9.0–10.0%. Regarding the required memory, note that approximately 0.6 MB of precomputation memory should be available to fully exploit Algorithm 5 when a 128-dimensional feature vector is used, i.e., $d = 130$. If multiple users are registered to the authentication server, the amount of required memory will be proportional to the number of registered users. For example, 600 MB is required to store the precomputation data for 1000 users. However, if we use Algorithm 6, we may permit more users, slightly degrading the authentication speed.

Table 3. Performance improvement in the facial verification system using the proposed method compared to the Naive method (times in ms).

Biometric Dataset	Face Image Processing [2]	EncodeY	Euclid			Total (Naive)	Total (Proposed)	Ratio
			Pairing (Naive)	Pairing (Proposed)	DLP			
Auto	193.27	106.47	58.86	15.66	120.30	478.90	435.70	9.0 %
Guide	157.47					443.10	399.90	9.7%
CFP	150.15					435.78	392.58	9.9%
ORL	147.33					432.96	389.76	10.0%

Security Analysis

To show that our FE-based facial verification system manages and processes the biometric information in a secure and privacy-compliant manner, we evaluate our system based on the requirements of biometric information protection, i.e., irreversibility, unlinkability, renewability, and performance [76,77].

- Irreversibility: the irreversibility of our system depends on the security of Π_{IPE}. Please note that the template is not stored in the client device. Therefore, the only concern is the possibility of template recovery on the server side. However, the Π_{IPE} guarantees that the server cannot learn any information about the stored ciphertext, except its inner product with another ciphertext. Therefore, the encrypted biometric information is protected by Π_{IPE}.
- Unlinkability: Our procedure for template encryption, i.e., $IPE.Encrypt$, involves a uniformly random component β. Consequently, it may produce completely different ciphertexts even when the same biometric information is encrypted. Therefore, it is not possible to link two or more biometric templates encrypted using Π_{IPE}.
- Renewability: Every call to the template encryption procedure $IPE.Encrypt$ generates a completely different ciphertext even for the same user using a random parameter β. Therefore, it can create multiple, independently transformed biometric templates.
- Performance: According to the ISO/IEC 19795-1 standard [77], we consider the biometric accuracy as a criterion of performance. It is straightforward that the accuracy of the proposed system is exactly the same as that of the underlying biometric verification system, because the proposed system does not revise the feature extraction process. It only encrypts the extracted features. Therefore, the biometric similarity score computed from $ct = EncodeX(msk, \mathbf{x})$ and $sk = EncodeY(msk, \mathbf{y})$ is exactly the same as that computed from the original $\mathbf{x}$ and $\mathbf{y}$.

Finally, we remark that the biometric information is still protected even when the precomputation on the biometric template is applied to the system. According to Algorithm 3, precomputation is performed only using s, m, and Q, where s and m are publicly known parameters. Q (Q_j in later algorithms) is an element in the encrypted biometric template vector. Therefore, when the server performs Algorithm 3 for precomputation, it does not obtain any additional information about the feature value encoded in the template. Consequently, whether the server performs the precomputation or not does not affect the security of the system. The same holds for the case where the server is compromised by an attacker.

7. Discussion

In this paper, we proposed a method to accelerate the inner-pairing product operation for secure biometric verification. The proposed Fixed-Q Product method is a method that optimizes inner-pairing product computation by applying precomputation. We also applied the new inner-pairing product method to design a secure biometric verification system. To verify the feasibility of the proposed method, we emulated a simple facial verification system comprising a client and a server. Our analysis results indicate that the new inner-pairing product method accelerates biometric authentication. However, the proposed method has a time-memory trade-off. The reduction in the amount of computation for the proposed method is obtained at the expense of memory to store the precomputed elements. Although the new method requires more memory than the previous methods, the amount of the additional memory is not considerable. Moreover, there are no changes in the bandwidth requirement for communication as the precomputation is performed on the server side. Furthermore, we also provide the adaptive method where the amount of precomputed elements is parameterized. Therefore, a server can choose to apply the proposed method by itself while a client is not aware of it. In other words, selectively tuning the performance of the system is possible.

We remark that noise may affect the security and performance of a biometric authentication system [78]. Therefore, we consider the effect of noise on the proposed method. First, note that the proposed system does not revise the feature extraction procedure, but it only encrypts the extracted features. Therefore, the noise-robustness of the underlying system is maintained even after applying our method, if only the integrity of a transmitted or stored ciphertext is guaranteed. On the contrary, if a ciphertext is modified, the server will be able to notice this change immediately, because the modification of even a single bit will make the data invalid. The probability that a modified element becomes a valid field

element constituting a ciphertext is almost zero. In summary, the proposed method does not affect the noise-robustness of a biometric authentication system.

Finally, the proposed method is suitable not only for remote biometric verification systems but also for any FE-based privacy-preserving applications where the evaluated function is an inner product and one of the two inputs of the inner product is entered offline [35,36]. Furthermore, the proposed method can be applied not only to FE-based systems but also to the other systems involving inner-pairing products, such as non-interactive zero-knowledge proofs [37,38].

Author Contributions: Conceptualization, M.-K.L.; Software, S.-Y.J.; Writing—original draft, S.-Y.J.; Writing—review & editing, M.-K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant 2020R1A2C2013089, and in part by the Inha University Research Grant (2020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

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

Abbreviations

The following abbreviations are used in this manuscript:

FE	Functional encryption
PIN	Personal identification number
EEG	Electroencephalography
HE	Homomorphic encryption
ZKPK	Zero-knowledge proof of knowledge
TEE	Trusted execution environment
BN curve	Barreto-Naehrig curve
IPE	Inner product encryption

References

1. *Information Technology—Vocabulary—Part 37: Biometrics*; Standard, International Organization for Standardization (ISO): Geneva, Switzerland, 2017.
2. Im, J.H.; Jeon, S.Y.; Lee, M.K. Practical Privacy-Preserving Face Authentication for Smartphones Secure Against Malicious Clients. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2386–2401. [CrossRef]
3. Jo, Y.H.; Jeon, S.Y.; Im, J.H.; Lee, M.K. Security analysis and improvement of fingerprint authentication for smartphones. *Mob. Inf. Syst.* **2016**, *2016*, 8973828. [CrossRef]
4. McGoldrick, L.K.; Halámek, J. Recent Advances in Noninvasive Biosensors for Forensics, Biometrics, and Cybersecurity. *Sensors* **2020**, *20*, 5974. [CrossRef] [PubMed]
5. Bollella, P.; Katz, E. Biosensors—Recent Advances and Future Challenges. *Sensors* **2020**, *20*, 6645. [CrossRef]
6. TrustZone—Arm Developer. Available online: https://developer.arm.com/ip-products/security-ip/trustzone (accessed on 31 January 2021).
7. Storing Keys in the Secure Enclave. Available online: https://developer.apple.com/documentation/security/certificate_key_and_trust_services/keys/storing_keys_in_the_secure_enclave (accessed on 31 January 2021).
8. Ratha, N.K.; Connell, J.H.; Bolle, R.M. Enhancing security and privacy in biometrics-based authentication systems. *IBM Syst. J.* **2001**, *40*, 614–634. [CrossRef]
9. Ratha, N.K.; Chikkerur, S.; Connell, J.H.; Bolle, R.M. Generating cancelable fingerprint templates. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 561–572. [CrossRef] [PubMed]
10. Juels, A.; Wattenberg, M. A fuzzy commitment scheme. In Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS '99), Singapore, 1–4 November 1999; pp. 28–36.
11. Quan, F.; Fei, S.; Anni, C.; Feifei, Z. Cracking cancelable fingerprint template of Ratha. In Proceedings of the 2008 International Symposium on Computer Science and Computational Technology (ISCSCT 2008), Shanghai, China, 20–22 December 2008; Volume 2, pp. 572–575.
12. Shin, S.W.; Lee, M.K.; Moon, D.; Moon, K. Dictionary attack on functional transform-based cancelable fingerprint templates. *ETRI J.* **2009**, *31*, 628–630. [CrossRef]

13. Nagar, A.; Nandakumar, K.; Jain, A.K. Biometric template transformation: A security analysis. In Proceedings of the Media Forensics and Security II. International Society for Optics and Photonics, San Jose, CA, USA, 27 January 2010; Volume 7541, p. 75410O.
14. Karimian, N.; Guo, Z.; Tehranipoor, M.; Forte, D. Highly reliable key generation from electrocardiogram (ECG). *IEEE Trans. Biomed. Eng.* **2016**, *64*, 1400–1411. [CrossRef] [PubMed]
15. Nguyen, D.; Tran, D.; Sharma, D.; Ma, W. On the study of EEG-based cryptographic key generation. *Procedia Comput. Sci.* **2017**, *112*, 936–945. [CrossRef]
16. Wang, P.; You, L.; Hu, G.; Hu, L.; Jian, Z.; Xing, C. Biometric key generation based on generated intervals and two-layer error correcting technique. *Pattern Recognit.* **2021**, *111*, 107733. [CrossRef]
17. Jalaly Bidgoly, A.; Jalaly Bidgoly, H.; Arezoumand, Z. A survey on methods and challenges in EEG based authentication. *Comput. Secur.* **2020**, *93*, 101788. [CrossRef]
18. Boyen, X.; Dodis, Y.; Katz, J.; Ostrovsky, R.; Smith, A. Secure remote authentication using biometric data. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 2005), Aarhus, Denmark, 22 May 2005; pp. 147–163.
19. Bhattasali, T.; Saeed, K.; Chaki, N.; Chaki, R. A survey of security and privacy issues for biometrics based remote authentication in cloud. In Proceedings of the International Conference on Computer Information Systems and Industrial Management (CISIM 2015), Warsaw, Poland, 24 September 2015; pp. 112–121.
20. Bringer, J.; Chabanne, H.; Patey, A. Privacy-preserving biometric identification using secure multiparty computation: An overview and recent trends. *IEEE Signal Process. Mag.* **2013**, *30*, 42–52. [CrossRef]
21. Rui, Z.; Yan, Z. A survey on biometric authentication: Toward secure and privacy-preserving identification. *IEEE Access* **2018**, *7*, 5994–6009. [CrossRef]
22. Kim, S.; Lewi, K.; Mandal, A.; Montgomery, H.; Roy, A.; Wu, D.J. Function-Hiding Inner Product Encryption is Practical. In Proceedings of the International Conference on Security and Cryptography for Networks (SCN 2018), Amalfi, Italy, 5 September 2018; pp. 544–562.
23. Zhou, K.; Ren, J. PassBio: Privacy-preserving user-centric biometric authentication. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 3050–3063. [CrossRef]
24. Lee, J.; Kim, D.; Kim, D.; Song, Y.; Shin, J.; Cheon, J.H. *Instant Privacy-Preserving Biometric Authentication for Hamming Distance*; Cryptology ePrint Archive, Report 2018/1214; IACR: 2018. Available online: https://eprint.iacr.org/2018/1214 (accessed on 1 April 2021).
25. Barbosa, M.; Catalano, D.; Soleimanian, A.; Warinschi, B. *Efficient Function-Hiding Functional Encryption: From Inner-Products to Orthogonality*; Cryptographers' Track at the RSA Conference (CT-RSA 2019); Springer: Berlin, Germany, 2019; pp. 127–148.
26. Zhao, Q.; Zeng, Q.; Liu, X. Improved Construction for Inner Product Functional Encryption. *Secur. Commun. Netw.* **2018**, *2018*, 6561418. [CrossRef]
27. Abdalla, M.; Bourse, F.; De Caro, A.; Pointcheval, D. Simple functional encryption schemes for inner products. In Proceedings of the IACR International Workshop on Public Key Cryptography (PKC 2015), Gaithersburg, MD, USA, 30 March–1 April 2015; pp. 733–751.
28. Datta, P.; Dutta, R.; Mukhopadhyay, S. Functional encryption for inner product with full function privacy. In Proceedings of the IACR International Workshop on Public Key Cryptography (PKC 2016), Taipei, Taiwan, 6–9 March 2016; pp. 164–195.
29. Kim, S.; Kim, J.; Seo, J.H. A new approach to practical function-private inner product encryption. *Theor. Comput. Sci.* **2019**, *783*, 22–40. [CrossRef]
30. Bünz, B.; Maller, M.; Mishra, P.; Tyagi, N.; Vesely, P. *Proofs for Inner Pairing Products and Applications*; Cryptology ePrint Archive, Report 2019/1177; IACR: 2019. Available online: https://eprint.iacr.org/2019/1177 (accessed on 1 April 2021).
31. Scott, M. Computing the Tate pairing. In Proceedings of the Cryptographers' Track at the RSA Conference (CT-RSA 2005), San Francisco, CA, USA, 14–18 February 2005; pp. 293–304.
32. Granger, R.; Smart, N.P. On Computing Products of Pairings. *IACR Cryptol. EPrint Arch.* **2006**, *2006*, 172.
33. Zavattoni, E.; Perez, L.J.D.; Mitsunari, S.; Sánchez-Ramırez, A.H.; Teruya, T.; Rodríguez-Henríquez, F. Software implementation of an attribute-based encryption scheme. *IEEE Trans. Comput.* **2014**, *64*, 1429–1441. [CrossRef]
34. Costello, C.; Stebila, D. Fixed argument pairings. In Proceedings of the International Conference on Cryptology and Information Security in Latin America (Latincrypt 2010), Puebla, Mexico, 8–11 August 2010; pp. 92–108.
35. Im, J.H.; Kwon, H.Y.; Jeon, S.Y.; Lee, M.K. Privacy-Preserving Electricity Billing System Using Functional Encryption. *Energies* **2019**, *12*, 1237. [CrossRef]
36. Son, Y.B.; Im, J.H.; Kwon, H.Y.; Jeon, S.Y.; Lee, M.K. Privacy-Preserving Peer-to-Peer Energy Trading in Blockchain-Enabled Smart Grids Using Functional Encryption. *Energies* **2020**, *13*, 1321. [CrossRef]
37. Anada, H. Decentralized Multi-authority Anonymous Authentication for Global Identities with Non-interactive Proofs. *J. Internet Serv. Inf. Secur.* **2020**, *10*, 23–37.
38. Pop, C.D.; Antal, M.; Cioara, T.; Anghel, I.; Salomie, I. Blockchain and Demand Response: Zero-Knowledge Proofs for Energy Transactions Privacy. *Sensors* **2020**, *20*, 5678. [CrossRef]

39. Chun, H.; Elmehdwi, Y.; Li, F.; Bhattacharya, P.; Jiang, W. Outsourceable two-party privacy-preserving biometric authentication. In Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, Kyoto, Japan, 4–6 June 2014; pp. 401–412.
40. Cheon, J.H.; Chung, H.; Kim, M.; Lee, K.W. *Ghostshell: Secure Biometric Authentication Using Integrity-Based Homomorphic Evaluations*; Cryptology ePrint Archive, Report 2016/484; IACR: 2016. Available online: https://eprint.iacr.org/2016/484 (accessed on 1 April 2021).
41. Im, J.; Choi, J.; Nyang, D.; Lee, M. Privacy-Preserving Palm Print Authentication Using Homomorphic Encryption. In Proceedings of the 2nd Int. Conf. Big Data Intell. Comput., Thessaloniki, Greece, 23–25 October 2016; pp. 878–881.
42. Lin, D.; Hilbert, N.; Storer, C.; Jiang, W.; Fan, J. UFace: Your universal password that no one can see. *Comput. Secur.* **2018**, 77, 627–641. [CrossRef]
43. Shahandashti, S.F.; Safavi-Naini, R.; Safa, N.A. Reconciling user privacy and implicit authentication for mobile devices. *Comput. Secur.* **2015**, *53*, 215–233. [CrossRef]
44. Šeděnka, J.; Govindarajan, S.; Gasti, P.; Balagani, K.S. Secure outsourced biometric authentication with performance evaluation on smartphones. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 384–396. [CrossRef]
45. Gasti, P.; Šeděnka, J.; Yang, Q.; Zhou, G.; Balagani, K.S. Secure, fast, and energy-efficient outsourced authentication for smartphones. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2556–2571. [CrossRef]
46. Abidin, A. On Privacy-Preserving Biometric Authentication. In Proceedings of the Information Security and Cryptology, Beijing, China, 29 November 2017; pp. 169–186.
47. Gunasinghe, H.; Bertino, E. PrivBioMTAuth: Privacy Preserving Biometrics-Based and User Centric Protocol for User Authentication From Mobile Phones. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1042–1057. [CrossRef]
48. Droandi, G.; Barni, M.; Lazzeretti, R.; Pignata, T. SEMBA:SEcure multi-biometric authentication. *arXiv* **2018**, arXiv:abs/1803.10758.
49. Catalano, D.; Fiore, D. Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015; pp. 1518–1529.
50. Damgård, I.; Pastro, V.; Smart, N.; Zakarias, S. Multiparty Computation from Somewhat Homomorphic Encryption. In Proceedings of the CRYPTO 2012, Barbara, CA, USA, 19–23 August 2012; pp. 643–662.
51. Sengupta, S.; Cheng, J.; Castillo, C.; Patel, V.; Chellappa, R.; Jacobs, D. Frontal to Profile Face Verification in the Wild. In Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 7–10 March 2016.
52. The Database of Faces (Formerly 'The ORL Database of Faces'). Available online: http://cam-orl.co.uk/facedatabase.html (accessed on 1 April 2021).
53. Taigman, Y.; Yang, M.; Ranzato, M.; Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 1701–1708.
54. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
55. Barreto, P.S.; Naehrig, M. Pairing-friendly elliptic curves of prime order. In Proceedings of the International Workshop on Selected Areas in Cryptography (SAC 2005), Kingston, ON, Canada, 11–12 August 2005; pp. 319–331.
56. Aranha, D.F.; Barreto, P.S.; Longa, P.; Ricardini, J.E. The realm of the pairings. In Proceedings of the International Conference on Selected Areas in Cryptography (SAC 2013), Burnaby, BC, Canada, 14–16 August 2013; pp. 3–25.
57. El Mrabet, N.; Joye, M. *Guide to Pairing-Based Cryptography*; CRC Press: Boca Raton, FL, USA, 2017.
58. Silverman, J.H. *The Arithmetic of Elliptic Curves*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2009; Volume 106.
59. Miller, V.S. The Weil pairing, and its efficient calculation. *J. Cryptol.* **2004**, *17*, 235–261. [CrossRef]
60. Scott, M.; Benger, N.; Charlemagne, M.; Perez, L.J.D.; Kachisa, E.J. On the final exponentiation for calculating pairings on ordinary elliptic curves. In Proceedings of the International Conference on Pairing-Based Cryptography (Pairing 2009), Palo Alto, CA, USA, 12–14 August 2009; pp. 78–88.
61. Cohen, H.; Frey, G.; Avanzi, R.; Doche, C.; Lange, T.; Nguyen, K.; Vercauteren, F. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*; CRC Press: Boca Raton, FL, USA, 2005.
62. Granger, R.; Hess, F.; Oyono, R.; Thériault, N.; Vercauteren, F. Ate pairing on hyperelliptic curves. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 2007), Barcelona, Spain, 20–24 May 2007; pp. 430–447.
63. Hess, F.; Smart, N.P.; Vercauteren, F. The eta pairing revisited. *IEEE Trans. Inf. Theory* **2006**, *52*, 4595–4602. [CrossRef]
64. Matsuda, S.; Kanayama, N.; Hess, F.; Okamoto, E. Optimised versions of the ate and twisted ate pairings. In Proceedings of the International Conference on Cryptography and Coding (IMACC 2007), Cirencester, UK, 18–20 December 2007; pp. 302–312.
65. Zhao, C.A.; Zhang, F.; Huang, J. A note on the Ate pairing. *Int. J. Inf. Secur.* **2008**, *7*, 379–382. [CrossRef]
66. Lee, E.; Lee, H.S.; Park, C.M. Efficient and generalized pairing computation on abelian varieties. *IEEE Trans. Inf. Theory* **2009**, *55*, 1793–1803. [CrossRef]
67. Vercauteren, F. Optimal pairings. *IEEE Trans. Inf. Theory* **2009**, *56*, 455–461. [CrossRef]

68. Beuchat, J.L.; González-Díaz, J.E.; Mitsunari, S.; Okamoto, E.; Rodríguez-Henríquez, F.; Teruya, T. High-speed software implementation of the optimal ate pairing over Barreto–Naehrig curves. In Proceedings of the International Conference on Pairing-Based Cryptography (Pairing 2010), Yamanaka Hot Spring, Japan, 13–15 December 2010; pp. 21–39.
69. Bishop, A.; Jain, A.; Kowalczyk, L. Function-hiding inner product encryption. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2015), Auckland, New Zealand, 29 November–3 December 2015; pp. 470–491.
70. Hankerson, D.; Menezes, A.J.; Vanstone, S. *Guide to Elliptic Curve Cryptography*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2006.
71. Jeon, S.Y.; Lee, M.K. Poster: Acceleration of Pairing Product Operation Using Precomputation. In Proceedings of the 21st World Conference on Information Security Applications 2020 (WISA 2020), Jeju Island, Korea, 26–28 August 2020; p. 5.
72. Jeon, S.Y. Acceleration of Pairing Operation for Performance Improvement of Functional Encryption. Master's Thesis, Inha University, Incheon, Korea, 2020.
73. GNU Multiple Precision Arithmetic Library (GMP). Available online: https://gmplib.org/ (accessed on 31 January 2021).
74. GitHub—Herumi/Mcl: A Portable and Fast Pairing-Based Cryptography Library. Available online: https://github.com/herumi/mcl (accessed on 31 January 2021).
75. A Library for Doing Number Theory (NTL). Available online: https://www.shoup.net/ntl/ (accessed on 31 January 2021).
76. *Information Technology—Security Techniques—Biometric Information Protection*; Standard, International Organization for Standardization (ISO): Geneva, Switzerland, 2011.
77. *Information Technology—Biometric Performance Testing and Reporting—Part 1: Principles and Framework*; Standard; International Organization for Standardization (ISO): Geneva, Switzerland, 2006.
78. Lafkih, M.; Mikram, M.; Ghouzali, S.; El Haziti, M. Evaluation of the Impact of Noise on Biometric Authentication Systems. In Proceedings of the 2019 3rd International Conference on Advances in Artificial Intelligence, Istanbul, Turkey, 26–28 October 2019; pp. 188–192.