Secure Information Processing with Privacy Assurance – Standard based Design and Development for Biometric Applications

Bon K. Sy, *Member, IEEE,* Adam Ramirez, Arun P. Kumara Krishnan

***Abstract*— This paper presents the design and development of a technique referred to as SIPPA – Secure Information Processing with Privacy Assurance – for biometric data reconstruction. SIPPA enables a client/server model with the following two properties: (1) the client party can compare the similarity between his/her sample data with the source data on the server side --- without each party revealing his/her data to another, nor to a third party. If the sample data is "sufficiently similar" to the source data, the client can reconstruct the source data by using only the sample data and some helper data with negligible overhead provided by the server. The main contributions of this paper are: (1) algorithmic steps of SIPPA and its relationship to privacy homomorphism, (2) a parallel SIPPA architecture, and (3) the realization of parallel SIPPA as a service component for BioAPI 2.0 framework using Java RMI technology. To demonstrate its potential application, we apply SIPPA to the reconstruction of biometric data, and more specifically, biometric face images represented in terms of linearized vectors.**

I. Introduction

Digital images are an essential medium for biometric data. Privacy safeguard on the communication of digital images with personal biometrics is an important factor for practical applications. In the United States when transmitting medical and health information, privacy and security rules are explicitly required by law under HIPPA [1] in the health care industry. Consider the following hypothetical scenario:

> *There is a database of medical records that contain face images of individuals with different frontal and side views. We would like to retrieve the face images of individuals collected over time and/or with a specific view similar to the one on hand for the purpose of studying aging.*

Traditionally, the retrieval and review process comprises two steps. First, an image is retrieved from the database. Then the image on hand is compared against the retrieved image. If there is a match, then further information about the retrieved image is extracted. The problem with this process is the inevitable privacy leak.

When the retrieved image does not match the image on hand, the private identity information associated with the image is leaked. In this research, we will focus on issues that can be formulated as below:

> There are two parties P1(Client) and P2 (Server). Party P1 and P2 have private data D1 and D2. Without the presence of a trusted third party, P1 and P2 would like to know whether D1 and D2 are sufficiently similar. And if so, P1 could derive D2 under the following two conditions:
> 1. P1 and P2 have to first find out whether D1 and D2 are sufficiently similar without any party sharing their private data to the another party.
> 2. If D1 and D2 are sufficiently similar, P1 can derive D2 without P2 ever sending D2. The only data that P2 will send to P1 is some helper data with negligible overhead.

SIPPA – Secure Information Processing with Privacy Assurance, is developed to achieve the comparison and retrieval tasks just mentioned. SIPPA is a kind of secure computation [2] grounded on privacy homomorphism [3]. The basic idea behind SIPPA is to model data (D1/D2) as a linearized vector, and to transform the vector into a symmetric matrix. To compare the two data sets, we take advantage of the approximately consistent relationship between the norm deviation of the data and the norm deviation of the corresponding eigenvalues/eigenvectors.

For privacy protection, neither party will share data, the corresponding matrix and the eigenvalues/eigenvectors with the other party. Instead, P1 and P2 will engage in a two-party secure computation for solving an algebraic system of linear equations. These linear equations define the constraint relationship between a separator boundary and the eigenvectors of the two parties; whereas the solution to the algebraic system reveals the information needed to derive the lower bound distance of the norm deviation of the eigenvectors. If the lower bound is deemed acceptable, the eigenvalue and some scalar value would be sent by the server (P2) to the client (P1) as helper data. Based on the helper data and D1, P1 can derive the norm deviation between the eigenvectors of P1 and P2 under the assumption of equi-distance. Subsequently, P1 can derive a sufficiently good approximation of D2. In this paper, we report the development of SIPPA, its implementation as a service component for BioAPI framework [21], and its application to face biometrics captured and stored in common digital image formats such as JPEG, PPM, and PGM [4].

In section II we will survey the current research on secure computation, and discuss lossless approach to protecting the privacy of personal biometrics. In section III the concept of privacy homomorphism is introduced. In section IV the theoretical formulation and the mathematical properties of

This work was supported in part by a grant from PSC-CUNY Research Award and NSF DUE 0837535. Bon K. Sy, Adam Ramirez, and Arun P. Kumara Krishnan are with the Computer Science Department, Queens College; and Bon K. Sy is also with University Graduate Center, City University of New York, 65-30 Kissena Blvd, Flushing NY 11367 USA (phone: 1-718-997-3566; email: bon@bunny.cs.qc.cuny.edu)

SIPPA will be presented and illustrated graphically. The algorithmic steps of SIPPA and parallel SIPPA architecture are discussed in sections V and VI. In section VII we describe the implementation of SIPPA as a service component using Java RMI technology, and the strategy towards exposing the service component in the standard based BioAPI framework. Experimental studies and the results will be reported in section VIII, followed by a conclusion in section IX.

II. Related Work

The privacy of personal biometrics is generally protected by means of two approaches; namely, lossy data processing and lossless data processing. In security surveillance, protecting privacy through lossy approach is not uncommon. Preserving privacy through the lossy approach is achieved by protecting the private content typically by means of perturbation, randomization or masking of the original data to the extent that it could still be useful for the security purposes [5][6]. In general, the degree of privacy protection based on lossy anonymization is data dependent and may not be extendable from one application to another that have different privacy requirements. For example, the k-Same model based on the k-anonymity framework proposed by Newton et al. [5] would not be able to adequately provide privacy protection when k is small and/or known unique aspect of an individual is not sufficiently anonymized.

An approach towards lossless privacy protection is Secure Multi-party Computation (SMC). SMC protects computational privacy while preserving content [7]. In other words, the original image content on personal biometrics can be retrieved and used in some secure computation scheme for deriving event information of interest, but the scope is limited to what is allowed by the private computational mechanism of the process.

Generally speaking, SMC deals with the problem in which multiple parties with private inputs would like to compute jointly some function of their inputs, but no party wishes to reveal its private input to other participants. For example, a physician (party P1) wants to compare a medical image of a patient containing personal biometrics with the medical image of another patient stored in a hospital database. The data custodian (e.g., party P2 who is the database administrator) in the hospital and the physician (party P1) may participate in a SMC protocol to jointly compute the output of a matching function that compares the personal biometrics in the medical images. The multi-party computation problem was first introduced by Yao [2] and extended by Goldreich et al. [8], and by many others.

Goldreich [8] pointed out that solutions to specific problems should be developed and customized for efficiency reasons. Du and Atallah [9, 10] proposed a protocol for PPCSC – referred to as Privacy-Preserving Cooperative Scientific Computations. PPCSC is a two-party secure computation problem whereas the two parties needs to solve x in (A1+A2)x = b1 +b2. In PPCSC, matrix A1 and vector b1 belong to party P1, matrix A2 and vector b2 belong to party P2. At the end of the PPCSC, both parties know the solution x while nobody knows the other party's private inputs.

In SIPPA, the private data exchange and information processing involves PPCSC. Specifically, the image containing personal biometrics is represented in the form of a matrix Ai (i=1,2), and the (most significant) eigenvector weighted by the eigenvalue is represented by the vector bi (i=1,2). We will show in the later section that the solution vector x satisfying (A1+A2)x = b1 +b2 for PPCSC offers the boundary information for each party to estimate the distance between the eigenvectors of the data matrices of the two parties. This distance estimate then forms the basis for comparing the images of both parties as well as the generation of helper data for reconstructing the desired image. Further details about this will be discussed in the later section.

In this research we tackle the problem of PPCSC different from Du's protocol that relies on the 1-out-of-N oblivious transfer protocol (1-n OT) [11,12]. Since 1-n OT could be computationally expensive [13], we instead employ homomorphic encryption and singular value decomposition (SVD) on the matrices of P1 and P2 to achieve privacy protection. Our approach is to take each private matrix and break it down into matrices through SVD, which gives us a partial view of the information needed for computing the biometric data. We then use SMC and homomorphic encryption to share the partial information between the participants in such a way that the original data can be reconstructed in the PPCSC without revealing any private information not intended for sharing. Further algorithmic details will be shown in section V.

III. Privacy Homomorphism

One of the key elements of SMC is homomorphic encryption. A classical example of homomorphic encryption scheme with multiplicative property is RSA [14]; i.e., given a RSA public key pk=(N,e) and cipher texts $\{c_i = m_i^e \bmod N\}$, the product of cipher texts $\Pi_i\ c_i = (\Pi_i\ m_i)^e \bmod N$. Similarly, encryption scheme with additive homomorphic property can be formulated as $\Pi_i\ c_i = Encrypt_S(pk, \sum_i m_i)$; whereas $Encrypt_S(pk, m_i)$ is the encryption of m_i using the public key pk under the scheme S. Exemplary encryption schemes with homomorphic additive property include Goldwasser-Micali [15], Benaloh [16], and Paillier [17].

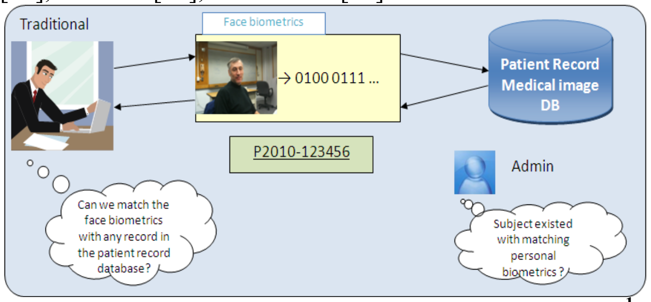

Traditional
Face biometrics
→ 0100 0111 ...
Patient Record Medical image DB
P2010-123456
Admin
Can we match the face biometrics with any record in the patient record database?
Subject existed with matching personal biometrics ?

Figure 1: Image Query without privacy protection[1]

Figure 1 further clarifies the concept of homomorphic encryption as applied to biometric data processing. In Figure 1, the image showing the face biometrics of an individual is

[1] The diagram is adopted from the one in a presentation entitled "Knowledge Emergence Infrastructure by Convergence of Real World and IT," by Akira Maeda, Systems development Laboratory, Hitachi, Ltd., Sept 30, 2009.

sent in plain to the remote database and is used for searching any matched images in the database. In this case, the biometric face information is unnecessarily exposed to the database administrator.

Figure 2 shows an image query based on homomorphic encryption for privacy protection. Instead of sending the image in plain, the encrypted version of the image is sent to the database administrator. The database administrator will search for a match by comparing the encrypted incoming image/face biometrics with the encrypted image/face biometrics in the database. Searching a match between two encrypted items is possible because of the mathematical property of the homomorphic encryption. Note that all images are encrypted and the comparison is performed without first decrypting the cipher image. Therefore, the privacy of the personal biometrics in an image is protected, and the database administrator does not know the face biometrics in the medical images as shown in Fig. 2.

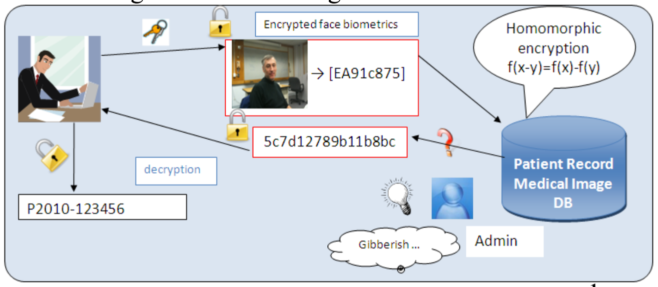

Encrypted face biometrics
→ [EA91c875]
Homomorphic encryption f(x-y)=f(x)-f(y)
5c7d12789b11b8bc
decryption
P2010-123456
Patient Record Medical Image DB
Gibberish ...
Admin

Figure 2: Image Query with privacy protection[1]

Although additive and multiplicative properties are two very desirable properties with a wide range of applications, these two properties are not sufficient to generalize all possible mathematical properties that may also be desirable for homomorphic encryption. Ideally, we would like to have full homomorphic encryption scheme as defined below:

Definition 1: Full homomorphic encryption scheme S with an efficient algorithm $Evaluate_S$ for any valid public key pk, any circuit C, and any cipher texts $\{c_i = Encrypt_S(pk, m_i)\}$ is defined as below:

$$Evaluate_S(pk, C, Encrypt_S(pk, m_1), \ldots, Encrypt_S(pk, m_t)) = Encrypt_S(pk, C(m_1, \ldots, m_t))$$

In other words, full homomorphic encryption scheme − excluding the trivial inefficient $Evaluate_S$ that is composed of decryption and C − allows one to arbitrarily compute on encrypted data. Full homomorphic encryption, also referred to as **complete privacy homomorphism**, is a significant classical problem introduced by Ron Rivest, Leonard Adleman and Michael Dertouzoz [18] almost 30 years ago. While progress has been made, it is only until recently that full homomorphic encryption is shown theoretically feasible using ideal lattice [3].

The basic idea behind the full homomorphic encryption based on ideal lattice is to first find a boots trappable circuit C that realizes an *approximate* homomorphic encryption scheme; i.e.; an encryption scheme that outputs a cipher text with small noise less than n, and a decryption scheme that can produce the original text so long as n is less than some threshold N >> n. Furthermore, there is a Recrypt algorithm taking a cipher text E(m) with noise N' < N, and producing a "fresh" cipher text output that encrypts m with a noise parameter $(N')^{0.5} < N$. In other words, the noise aggregate accumulated from the operation(s) over the cipher texts could be reduced recursively by calling Recrypt algorithm until the noise parameter converges to the range N permitting a valid decryption, thus preserving the properties and structure required for encryption and decryption. The *approximate* homomorphic encryption scheme is "boots trappable" if it is sufficient to derive the Recrypt algorithm.

The theoretical feasibility of privacy homomorphism has generated excitement in the research community because it opens the door for many potential practical applications. Realizing full homomorphic encryption scheme for practical applications such as private biometric information retrieval, exchange, and processing, however, is not trivial. As reported elsewhere [3], the theoretical boundary for the multiplicative depth of a decryption circuit is log log N – log log n -1. In order to cover a wide range of decryption circuits, N tends to be large while n tends to be small. Even in the log log space, the complexity of a typical decryption circuit based on ideal lattice is high, thus inefficient when it is compared to schemes such as RSA or ElGamal. Furthermore, it is still largely unknown the convergence rate of the Recrypt in practical applications. In order to balance the practicality and the extensibility of homomorphic encryption, we investigate a special homomorphic scheme referred to as <u>S</u>ecure <u>I</u>nformation <u>P</u>rocessing with <u>P</u>rivacy <u>A</u>ssurance.

SIPPA is not a fully homomorphic encryption scheme. Rather, SIPPA is grounded on the notion of sufficient privacy homomorphism. In other words, SIPPA does not compute on encrypted data using any arbitrary function. But SIPPA aims for computationally efficient and is general enough by requiring only the problem of privacy preserving to be linearly decomposable. We will show in the next section how the linearly decomposable property can be used for performing image comparison without exposing the personal biometrics in the image.

IV. Private & Secure Information Processing (SIPPA)

In this research the privacy model for biometrics can be formulated as below: Party P1 has an image expressed in terms of a linearized vector D1. Party P2 has some linearized vector template D2 about a subject P3. The objective is for P1 to determine whether D1 and D2 are similar under the following conditions:

1. P1 and P2 do not reveal their private data to each other.
2. P1 and P2 both need to determine whether D1 and D2 are *sufficiently similar*.
3. If D1 and D2 are *sufficiently similar*, P2 will provide some *helper data* HD with a negligible overhead for P1 to reconstruct D2 using only D1 and HD.

Eigen-based approach has been developed in the early 90s for people identification based on face biometrics [19]. An important property of eigenface is to represent a linearized image as a covariance matrix with its corresponding eigenvectors as a set of linearly independent basis for capturing the variation of the image.

Covariance matrix of an image representation is symmetric. In fact, any linearized data vector D will yield a symmetric matrix out of $D \cdot D^T$. Let A1 and A2 be the proper transformation of some linearized data D1 and D2 through $D1 \cdot D1^T$ and $D2 \cdot D2^T$ respectively. Let $\{(\lambda^1_i, V^1_i) | i = 1,2\ldots\}$ and $\{(\lambda^2_i, V^2_i) | i = 1,2\ldots\}$ be the corresponding sets of eigenvalues and unity normalized eigenvectors for A1 and A2 respectively. For the sake of discussion and without the loss of generality, we will focus on only the largest eigen components (λ^1_1, V^1_1) and (λ^2_1, V^2_1). Let x be a vector such that $(A1+A2)x = \lambda^1_1 \cdot V^1_1 + \lambda^2_1 \cdot V^2_1$. By definition, $A1V^1_1 = \lambda^1_1 \cdot V^1_1$ and $A2V^2_1 = \lambda^2_1 \cdot V^2_1$. These relationships manifest an endomorphism mapping with the following three observations:

Observation 1: $\lambda^1_1 \cdot V^1_1$ and $\lambda^2_1 \cdot V^2_1$ are the transformation of the eigenvectors V^1_1 and V^2_1 through A1 and A2 respectively.

Observation 2: The resultant sum of the vectors $\lambda^1_1 \cdot V^1_1$ and $\lambda^2_1 \cdot V^2_1$ are the transformation of the vector x through (A1+A2).

Observation 3: Vector x can be decomposed into components with V^i_1 (i=1,2) as basis; i.e., $x = V^1_1 + \varepsilon_1$ and $x = V^2_1 - \varepsilon_2$; whereas ε_i (i=1,2) can be considered as an error/offset term accounting for the deviation of x from V^i_1 (i=1,2).

The graphical interpretation of the observations just mentioned above is illustrated below:

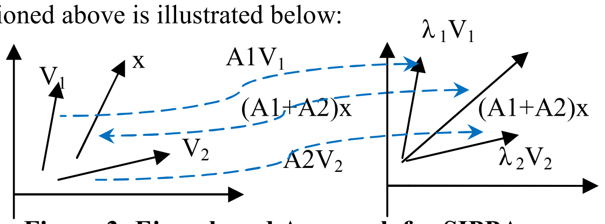


Figure 3: Eigen-based Approach for SIPPA

As one could notice from above, x converges to the eigenvector as (λ^1_1, V^1_1) and (λ^2_1, V^2_1) converge to each other. This, together with observation 3, lead to the following trivial, yet important property:

Property 1: As (λ^1_1, V^1_1) and (λ^2_1, V^2_1) converge to each other, x converges to the eigenvector V^i_1 (i=1,2) and ε_is (i=1,2) converge to zero.

The property above essentially states that one could infer the closeness between V^1_1 and V^2_1 through the V^i_1 and x without knowing V^j_1– the basis of SIPPA; where (i j) = (1 2) or (2 1). Furthermore, it can easily be shown that $A1\varepsilon_1 = A2\varepsilon_2$, or $D1 \cdot D1^T \cdot \varepsilon_1 = D2 \cdot D2^T \cdot \varepsilon_2$, which provides a convenient way in the SIPPA scheme to derive Dj when the scalar $Dj^T \cdot \varepsilon_j$ is known. In other words, SIPPA can easily facilitate "separation of duty" in the sense that data exchange/processing is only possible when both parties collaborate. Similarly, SIPPA separates the step for similarity comparison and that for retrieval. Therefore, the security principles "need-to-know" and "least privilege" can be implemented in the SIPPA environment.

V. SIPPA ALGORITHMIC DETAILS & COMPLEXITY ANALYSIS

There are three major aspects of SIPPA: (1) derivation of the eigenvalues and the corresponding unity normalized eigenvectors of the symmetric matrix representation of the data; (2) derivation of a boundary vector separating the eigenvectors of the two parties, which is formulated as a two-party PPCSC secure computation, and (3) reconstruction of the source data based on the helper data composed of the eigenvalue and a scalar derived from the vector product between the transpose of the linearized source data vector and the boundary vector. We will first outline the key steps of SIPPA, and then the secure computation protocol for PPCSC.

Let Dv and De be the sample and source linearized data respectively. The key steps of SIPPA are summarized below:

Step 1: Derive symmetric matrix representation of the data in the form of $Dv \bullet Dv^T$ (= A1) and $De \bullet De^T$ (= A2).

Step 2: Derive the eigenvalues and the corresponding unity normalized eigenvectors $\{(\lambda^1_i, V^1_i) | i = 1,2\ldots\}$ of $Dv \bullet Dv^T$ and $\{(\lambda^2_i, V^2_i) | i = 1,2\ldots\}$ of $De \bullet De^T$.

Step 3: Compute x such that $(A1+A2)x = \lambda^1_1 \cdot V^1_1 + \lambda^2_1 \cdot V^2_1$.

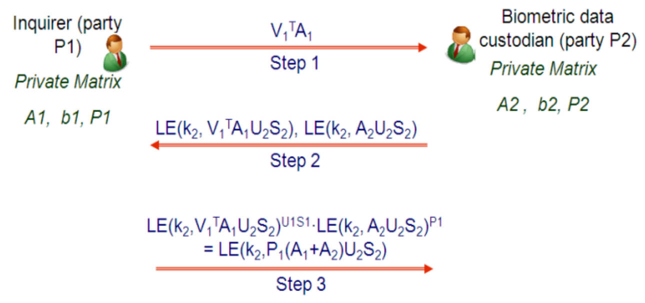
Step 4: Derive the closeness between De and Dv via the min. distance between V^1_1 and V^2_1. The min. distance between V^1_1 and V^2_1 is estimated via $||V^2_1 - x||$.

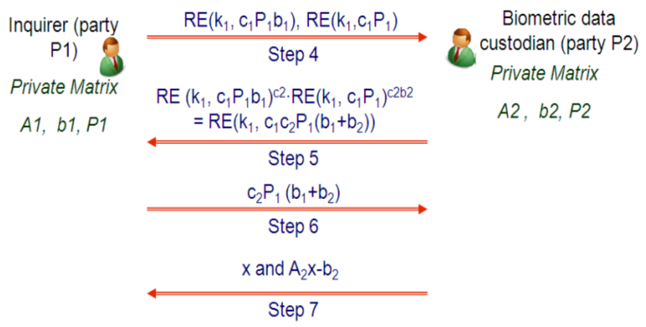
Step 5: If De and Dv are sufficiently close as measured by $||V^2_1 - x||$ < some pre-defined threshold, proceed to send the following helper data: λ^2_1 and $De^T \cdot x$.

Step 6: Derive $\underline{V}^2_1 = V^1_1 + 2(x - V^1_1)$, and then the estimated source data De□ = $\lambda^2_1 \cdot \underline{V}^2_1 / De^T \cdot x$

The boundary vector x in step 3 will be derived under PPCSC. The basic idea behind the secure computation protocol for PPCSC required in step 3 for solving $(A_1+A_2)x = b_1 + b_2$ is to solve $P_1(A_1+A_2)P_2y = P_1(b_1+b_2)$; where (P_1, A_1, b_1) are private to P1, and (P_2, A_2, b_2) are private to P2. Note that even if P2 knows $P_1(A_1+A_2)P_2$ and $P_1(b_1+b_2)$, P2 can only derive y but could not know A_1, b_1, and P_1; thus the privacy of A_1, b_1, and P_1 for P1 is preserved. Once y is solved, P2 can derive $x = P_2y$ and sends it to P1. Note that any adversary intercepting or sniffing from the network the value of x cannot derive De or Dv unless the adversary also has either (A_1, b_1, V^2_1) or (A_2, b_2, V^1_1).

During the process of secure computation, A_1, b_1, A_2, b_2 are never exposed individually except $P_1(A_1+A_2)P_2$ and $P_1(b_1+b_2)$ for P2. Therefore, it is information-theoretic secure [20]; i.e., the privacy of (A_1, b_1) for P1, and the privacy of (A_2, b_2) for P2, is guaranteed. The step-by-step details on the secure computation mechanism for solving x in $(A_1+A_2)x = b_1+b_2$ is shown below; where the singular value decomposition of $P_1 = U_1S_1V_1^T$, $P_2 = U_2S_2V_2^T$, k_1 and k_2 are homomorphic encryption keys, c_1 and c_2 are some random numbers, and LE(●) and RE(●) represent left and right homomorphic encryption respectively. Further details can be found at [7].

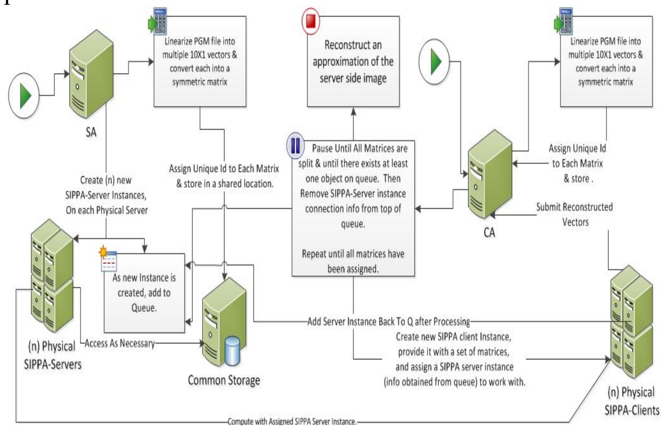

Inquirer (party P1)
Private Matrix
A1, b1, P1
$V_1^TA_1$
Step 1
Biometric data custodian (party P2)
Private Matrix
A2, b2, P2
$LE(k_2, V_1^TA_1U_2S_2)$, $LE(k_2, A_2U_2S_2)$
Step 2
$LE(k_2, V_1^TA_1U_2S_2)^{U1S1} \cdot LE(k_2, A_2U_2S_2)^{P1}$
$= LE(k_2, P_1(A_1+A_2)U_2S_2)$
Step 3


Inquirer (party P1)
Private Matrix
A1, b1, P1
$RE(k_1, c_1P_1b_1), RE(k_1,c_1P_1)$
Step 4
$RE (k_1, c_1P_1b_1)^{c2}\cdot RE(k_1, c_1P_1)^{c2b2} = RE(k_1, c_1c_2P_1(b_1+b_2))$
Step 5
$c_2P_1 (b_1+b_2)$
Step 6
x and A_2x-b_2
Step 7
Biometric data custodian (party P2)
Private Matrix
A2 , b2, P2

In regard to the complexity of SIPPA, it can be accounted for by four main tasks; namely, matrix composition (step 1), eigenvector/eigenvalue discovery (step 2), singular value decomposition (step 3), and Euclidean distance calculation (step 4). Matrix composition is in the order of $O(n^2)$; where n is the dimension of data vector De/Dv. Eigenvector/ eigenvalue discovery relies on Eigendecomposition algorithm that has a complexity roughly in the order of $O(n^3)$. Singular value decomposition has a complexity in the order of $O(n^2r)$; where r is the rank of a matrix. The complexity of calculating the Euclidean distance is in the order of $O(n^2)$. In terms of real time performance, opportunity exists to parallelize SIPPA to handle large data vector (i.e., large n; e.g., order of 10^4).

VI. PARALLEL SIPPA

The essential idea behind Parallel SIPPA is the creation of multiple SIPPA-Client and SIPPA-Server instances. The parallel SIPPA architecture is summarized below:


Linearize PGM file into multiple 10X1 vectors & convert each into a symmetric matrix
Reconstruct an approximation of the server side image
Linearize PGM file into multiple 10X1 vectors & convert each into a symmetric matrix
SA
Create (n) new SIPPA-Server Instances, On each Physical Server
Assign Unique Id to Each Matrix & store in a shared location.
Pause Until All Matrices are split & until there exists at least one object on queue. Then Remove SIPPA-Server instance connection info from top of queue.
Repeat until all matrices have been assigned.
Assign Unique Id to Each Matrix & store.
Submit Reconstructed Vectors
CA
As new Instance is created, add to Queue.
(n) Physical SIPPA-Servers
Access As Necessary
Common Storage
Add Server Instance Back To Q after Processing
Create new SIPPA client Instance, provide it with a set of matrices, and assign a SIPPA server instance (info obtained from queue) to work with.
(n) Physical SIPPA-Clients
Compute with Assigned SIPPA Server Instance.
Figure 4: Parallel SIPPA system delineation

SIPPA realized under parallel computing can produce faster results. In parallel SIPPA, a Server Application (SA) initializes multiple instances of SIPPA-Server in different physical/virtual machines. As SIPPA-Server instances are created, their contact information is added to a queue maintained by SA. In dealing with digital images in PGM/PPM file format, SA linearizes the file into multiple 10X1 vectors where each vector is used to produce a symmetric matrix described in step 1, section V. All SIPPA-Server instances have access to any of these symmetric matrices based on a matrix's unique id. When the Client Application (CA) starts, it takes possession of a sample client PGM/PPM image and linearizes the file into multiple 10X1 vectors where each vector is used to produce a symmetric matrix described in step 1, section V.

After CA contacts SA, SA de-queues an element and sends the contact information of SIPPA-Server instance to CA. CA continuously creates a new SIPPA-Client instance on available SIPPA-Server instances until all 10X1 vectors have been assigned resulting in parallel processing. CA provides every SIPPA-Client instance with an array of symmetric matrices and a chosen SIPPA-Server instance's contact information. This SIPPA-Client instance and its assigned SIPPA-server instance now engage in a sequential SIPPA session, processing each symmetric matrix at a time. As the SIPPA-Server/Client pairs finish processing their allotted set of matrices, the SIPPA-Server instances are added back to SA's queue and the SIPPA-Client instance is destroyed.

The reconstructed 10X1 vector produced by each SIPPA-client/SIPPA-Server pair is sent back to CA. When the entire set of 10X1 vectors is complete, all these vectors are combined to reconstruct an approximation of the server side image by CA.

VII. BIOAPI STANDARD BASED SIPPA IMPLEMENTATION

When job requests are presented to Parallel SIPPA (PSIPPA) one at a time, PSIPPA can achieve near real time performance as shown in the experiment results in section VIII. While the near real time results are noteworthy, our aim is to expand PSIPPA towards a true interoperable system. To achieve this we integrate PSIPPA into a BioAPI system which results in BioSIPPA. By integrating PSIPPA into a BioAPI 2.0 framework [21], PSIPPA is made available in an interoperable environment allowing multiple users on various platforms and operating systems to access PSIPPA through standard based service component. In order for BioSIPPA to handle simultaneous job requests from different users, we also investigate a Slice Based Architecture (SBA) design to optimize the resource utilization of BioSIPPA.

At the BioAPI level, realizing PSIPPA as a BioAPI service component involves a design and structural definition for various BioAPI framework components. At the minimum, we need to determine the appropriate level of PSIPPA instantiation in the BioAPI framework, and to define the corresponding framework components including Biometric Service Providers (BSP), Biometric Function Providers (BFP), and various BioAPI Units managed by the BSP or BFP such as Processing Algorithm Units (PAU) or Archive Units (AU). As defined in the BioAPI 2.0 standard, the PSIPPA schema and access interfaces are registered with the BioAPI component registry. This allows end users to search for our service.

Our targeted SBA-BioSIPPA system is composed of various elements that are the basis for its interoperable environment. Specific to biometric system development, the most atomic element in the SBA proposed in this project is resource. A resource could be a Java functional method exposed for an external call via Java RMI, a biometric software/hardware interface or a software/hardware processing unit for vector processing. A collection of resources for the purpose of completing a specific task (e.g., enrollment) is referred to as an aggregate. An aggregate only defines the set of resources; it does not define how the resources of an aggregate should be used. The concept of a service can be thought of as one or more aggregates with additional information on how the resources may be used to accomplish a specific task. Lastly, a service slice is essentially one or more aggregates provisioned by a slice manager. A slice manager coordinates how these aggregates may be utilized; e.g., by whom, for how long, under what pre-conditions or constraints on bandwidth, memory, CPU, or storage size. In the application of SBA-BioSIPPA, the slice manager drives the provisioning of slices which creates the necessary element for a true parallel and interoperable environment.

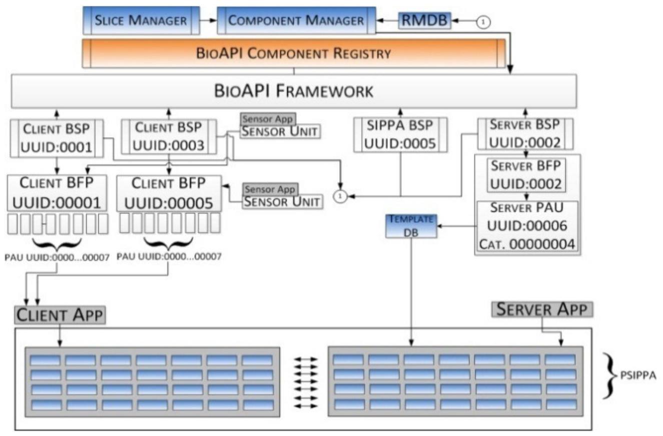


Figure 5: SIPPA Deployment for BioAPI Framework

An objective of SBA-BioSIPPA is to expose PSIPPA as an interoperable service component for a BioAPI 2.0 framework [21]. As a long term goal, we also aim for an easy adaptation of the service component for the standards developed for other application domains; e.g., ISO 19092:2008. Our strategy towards staging the PSIPPA service component for flexibility and adaptation is to employ Java RMI technology, which provides a lightweight, multithreaded, reliable communication bridge between the BioAPI and PSIPPA systems.

As in most biometric systems, SFA-BioSIPPA utilizes various resources similar to a face recognition system that has a Sensor Unit, a Matching Unit, and an Archive Unit. In SFA-BioSIPPA each necessary resource may be supplied by a different vendor and we provide a means for an end user to choose among the various resources.

To illustrate the utilization of SFA-BioSIPPA, consider a user A requests a verification job. User A will interact with the Slice Manager, which queries the Component Manager. The slice manager then displays what resources are available to execute the job request. User A, with the assistance of the Slice Manager, chooses the appropriate resources for the job and sends a request for a slice allocation. The request is recorded in an internal registry within the Slice Manager and then sent to the Component Manager. The Component Manager accesses the Resource Management Database (RMDB) and acquires the details of resources specific to the slice request. The resources are accessed through their BSP, at which point the slice is in its initiation stage. When a resource is accessed through its BSP, the BSP will initiate execution and immediately update a record in the RMDB incrementing the number of jobs executing on the particular resource. On the termination of a job on each resource, the resource's BSP will again update the database decrementing the number of jobs currently running on the resource. Any changes in the RMDB will allow future queries from the Slice Manager to reflect a near real time status of the system.

VIII. EXPERIMENTAL STUDY AND RESULT

The first part of our experimental study is to investigate the behavior of SIPPA from the following perspective: How is (1) the quality of the estimate on the closeness between De and Dv, and (2) the closeness between De and De□ (estimated De), affected by:

(a) Directional deviation between V^1_1 and V^2_1

(b) Dimension of x (thus De and Dv)

We generated 5 sets of simulated test data categorized by different dimensions. The vector dimensions in these five data sets are 5, 10, 20, 40, and 60 respectively. In each data set, 10 pairs of client (Dv)/server (De) vectors are generated, thus totaling 50 pairs for all dimensions. Due to the page limit, we only show the variation in the normalized error rate of the reconstruction |De- De□ |/Dim as a function of the data deviation |De-Dv|/Dim in Figure 6 below:

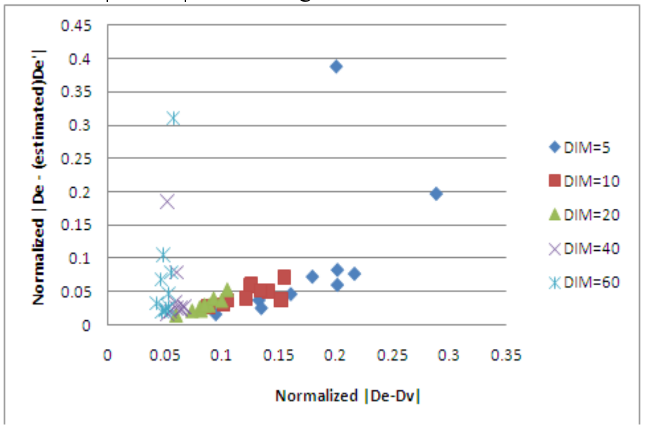


Figure 6: Estimation error of private data reconstruction

In the second part of our experimental study, we applied SIPPA to privately reconstruct an image containing biometric face information. From the result of the simulation study, we chose the data dimension to be 10. On the server side, there is a black-and-white image with 255 gray levels (Figure 7). The resolution of the image is 256x192. The image is stored in PGM (Portable Gray Map) format. The linearization of the image results in a 49152 (=256x192) x 1 vector. Obviously the dimension of the linearized vector is beyond the optimal performance range of SIPPA. As such, the 49152x1 linearized image vector is split into multiple 10x1 vectors (roughly 5000). SIPPA is then applied iteratively to reconstruct the original image.

During the reconstruction, two parties − referred to as client and server − will participate in a SIPPA session. The client has an image, referred to as a sample image. The server has an image, referred to as a source image. In the SIPPA session, each party takes a portion of the linearized image sequentially in the form of a 10x1 vector to construct a symmetric matrix, and derives the eigenvalue/eigenvector of the matrix. Then both parties participate in a secure computation protocol for PPCSC as described in step 3 of SIPPA in section V. The outcome of PPCSC is the boundary vector x. The client party then uses the boundary vector x and the client side 10x1 linearized vector of the sample image to reconstruct the server side 10x1 linearized vector of the source image. The only information that the server provides is the helper data composed of the eigenvalue of the server side 10x1 linearized source image and a scalar. The original 10x1 linearized source image is never shared with the client party. Once the reconstruction of the 10x1 server side linearized source image is completed, the SIPPA process repeats for the next block of the 10x1 linearized vector until all 10x1 image blocks are processed. The entire source image is then reconstructed by the client party using the estimated 10x1 source image blocks.

The SIPPA reconstruction of the entire source image is conducted twice in this experiment. In the first trial, the client side provides a sample image shown in Figure 8 that is sufficiently similar to the server side source image as shown in Figure 7. All images have a gray scale of 255 and have the same resolution 256x192.

The outcome of the reconstruction by applying SIPPA based on the server side source image (Figure 7) and the client side sample image (Figure 8) is shown in Figure 9. During the reconstruction, the source image is never shared with any party except the corresponding eigenvector information used in the SIPPA processing.

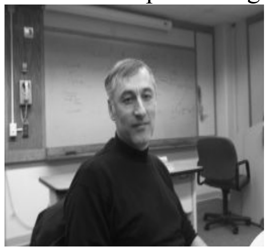


Figure 7: Source image

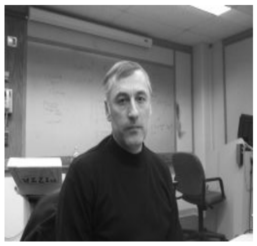


Fig. 8: Similar sample image

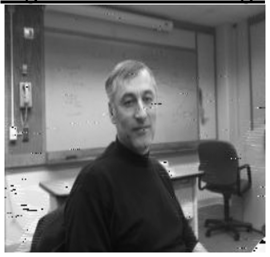


Fig. 9: Reconstruction using a similar image

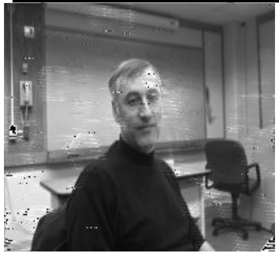


Fig. 10: Reconstruction using a dissimilar image

During the second trial, a sample image of a different human subject (not Figure 8) is used on the client side for SIPPA. The image of the other human subject is not shown in this paper due to the restriction on the human subject clearance. The outcome of the reconstruction is shown in Figure 10, which shows a poorer quality in comparison to Figure 9. By visual inspection and using Figure 7 as a reference, the face biometrics in Figure 9 is preserved better than that in Figure 10 – when the client side sample image is closer to that of the server side source image (Figure 7). In this experimentation, the threshold is set at infinity for step 5 of SIPPA described in section V. In other words, SIPPA will always proceed to step 6 for the reconstruction no matter the source/sample images are sufficiently similar or not. In real world practice, the threshold should be set at a reasonable value to reflect the tolerance on privacy risk.

In the last experimental study we try to determine the optimal configuration for Parallel SIPPA i.e. the number of physical/virtual machines that need to be assigned for SIPPA-Servers and SIPPA-Clients respectively and the number of SIPPA-Server instances that need to be created initially by SA (Server Application).

Performance of the SIPPA system is highly dependent on the degree of parallelism that can be obtained with the available hardware under which SIPPA operates. In our tests, we experiment with various platforms, both sequential and parallel; to pinpoint increases in performance as well as where our system is bottle-necked. By sequential we mean one instance of SIPPA-Server and SIPPA-Client runs at any given time in the entire system, and the 10X1 vectors are processed one at a time, whereas by parallel we mean multiple instances of SIPPA-Server and SIPPA-Client run at any given time processing multiple 10X1 vectors at any given time. Our testing configurations included: Single Core virtual machines (VM), Dual Core physical machines (DC), and a Quad Core machine (QC).

To obtain 90 - 100% CPU utilization on all physical/virtual machines (both server and client), a proper client to server ratio must be applied. This is because SIPPA-client instances have an additional workload when compared with SIPPA-Server instances; i.e. they obtain their matrices dynamically from CA and then store them locally for processing. This additional network I/O coupled with the fact that they also have to reconstruct the server side matrix makes the Client Machines consume 6 times more resources than a Server machine. With a naïve parallel approach of using one physical/virtual machine as a server for every physical/virtual client machine, performance is improved in comparison to a sequential SIPPA approach, but as we increase the client to server ratio by decreasing the amount of servers, the servers are more efficiently used which results in a dramatic decrease in completion time and full utilization of the CPU.

The data shows where our performance increase lies and where possible bottlenecks are. As seen in Figure 11, even when parallel SIPPA is run on a single quad core machine, a performance increase of 1600% is obtained when compared with using Two Dual Core machines engaging in SIPPA sequentially. Also notable is the increase between the naïve parallel approach and an optimized parallel approach. With the VMs we had an increase of 75% in performance when comparing a naïve parallel approach to an optimized parallel approach. With the large data set experiment on the physical machines, we noticed an increase of 350%.

During Parallel SIPPA startup, there is an initialization and job distribution phase; this seems to take anywhere between 30 and 60 seconds depending on size of image. Due to this, processing small images with a resolution 256x192 by parallel SIPPA do not seem to benefit much from increasing the number of physical/virtual machines within a certain setup.

In parallel SIPPA when an optimized setup is doubled and presented with an image of high resolution, we notice a reduction of processing time by about 40%; i.e., when the total number of machines is doubled from 8 (# 5) to 16 (# 6), the processing time for a 640X480 image decreases from 26min to 15 min. We observe a similar improvement in the case with virtual machines (# 8 and # 9). Doubling of resources reduces processing time by approximately 40%. This indicates that given a reasonable size cluster of professional level servers, most of the typical biometric images could be processed in a Parallel SIPPA framework within a reasonable amount of time.

#	Machine Setup
1	Two DC's (Dual Core-3.2GHz, 2.75GB RAM).
2	Two VM-512's.(Virtual Machine-1GHz,.5G RAM).
3	One QC. (Quad Core Machine-3.2GHz, 4G RAM).
4	Eight DC's Running multiple SS(SIPPA-Server) Instances and Eight DC's Running multiple SC(SIPPA-Client) instances.
5	Five VM-512's Running multiple SS Instances and Five VM-512's Running multiple SC instances.
6	16 (2 used for SS and 14 used for SC) DC's Running Multiple Instances Of SS and SC.
7	8 (1 used for SS and 7 used for SC) DC's Running Multiple Instances Of SS and SC.
8	10 (2 used for SS and 8 used for SC) VM-512's Running Multiple Instances Of SS and SC.
9	10 (2 used for SS and 8 used for SC) VM-1G's Running Multiple Instances Of SS and SC.
10	5 (1 used for SS and 4 used for SC) VM-1G's Running Multiple Instances Of SS and SC.
Engaging in sequential SIPPA. Optimized Parallel SIPPA where all CPU's are maintained at 100% usage for most of the time. Naïve Parallel SIPPA, Equal number of Physical/Virtual Machines For SS & SC.	

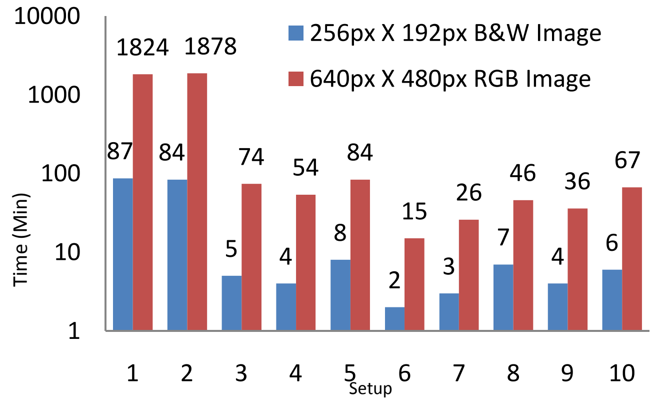


Figure 11: Parallel SIPPA processing time performance

IX. CONCLUSION

This paper presents a secure computation technique that guarantees security and privacy for the reconstruction of images containing biometric face information. Our secure computation technique, referred to as Secure Information Processing with Privacy Assurance (SIPPA), aims for sufficient privacy homomorphism and computational efficiency. Although it is not complete privacy homomorphism, SIPPA can be applied to the class of secure computation problems that is linearly decomposable. For proof of concept, we conducted both a simulation study and an application of SIPPA to reconstruct images with face biometrics. Our experimental study showed that the results were consistent with the theory behind SIPPA. To achieve near real time performance, parallel SIPPA is required. In dealing with digital images, SIPPA has to handle the computational load that increases in (1) the quadratic order of the image resolution, and (2) the choice of color quality and gray scale level. Two areas of future research of our particular interest to be focused on are: (1) a practical slice-based architecture to realize the full potential of SIPPA for real time performance, and (2) an extension of SIPPA that relaxes the restriction on the initial comparison of the client side image and the server side image from one-to-one to one-to-many.

REFERENCES

[1] http://www.hhs.gov/ocr/privacy/
[2] A.C.Yao., "Protocols for secure computations," In *23rd IEEE Sym. on Foundations of Computer Science* (1982).
[3] C. Gentry. "Fully Homomorphic Encryption Using Ideal Lattices," In the 41st ACM Symposium on Theory of Computing (STOC), 2009.
[4] http://en.wikipedia.org/wiki/Netpbm_format
[5] E. Newton, L. Sweeney, B. Mali, "Preserving Privacy by De-identifying Facial Images," IEEE Transactions on Knowledge and Data Engineering, 17 (2) February 2005, pp. 232-243.
[6] R. Gross, E. Airoldi, B. Malin, L. Sweeney, Integrating Utility into Face De-Identification. 2005
[7] B.K. Sy, "Secure Computation for Privacy Preserving Biometric Data Retrieval and Authentication," Euro ISI 2008, Esbjerg, Denmark, LNCS, Springer.
[8] Goldreich, O.: Secure Multi-Party Computation (working draft),
[9] W. Du, M.J. Atallah, "Privacy-Preserving Cooperative Scientific Computations." In *14th IEEE Computer Security Foundations Workshop*, pp. 273-282 (2001)
[10] W. Du, M.J. Atallah, "Secure Multi-Party Computation Problems and Their Applications: A Review and Open Problems." *In New Security Paradigms Workshop*, pp. 11-20 (2001)
[11] G. Brassard, C. Crepeau, J. Robert, "All-or-Nothing Disclosure of Secrets," Advances *in Cryptology-Crypto86, Lecture Notes in Computer Science*, pp. 234-238 (1987)
[12] S. Evan, O. Goldreich, Lempel, A.: "A Randomized Protocol for Signing Contracts." *Communications of the ACM*, vol. 28, 637-647 (1985)
[13] M. Naor, B. Pinkas, "Efficient Oblivious Transfer Protocols", In *20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 448-457, Washington D.C. (2001)
[14] R. Rivest, A. Shamir and L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, 21 (2), pp. 120-126, February 1978.
[15] Goldwasser S. and Micali S., "Probabilistic Encryption," *Journal of Computer and System Sciences (JCSS)*, 28(2):270-299, April 1984. Preliminary version in: 14th Annual ACM Symposium on Theory of Computing (STOC)
[16] J. Benaloh, "Dense Probabilistic Encryption," *Proceedings of the Workshop on Selected Areas of Cryptography*. Kingston, ON. May 1994. pp. 120--128.
[17] P. Paillier P., "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," EUROCRYPT 1999, pp223-238.
[18] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. In Foundations of Secure Computation, 1978.
[19] Turk and A. Pentland (1991). "Face recognition using eigenfaces". Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–591. Available online at: http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf.
[20] T. P. Pedersen, « Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, » LNCS Vol. 576, pp 129-140, 1991, ISBN : 3-540-55188-3.
[21] http://www.bioapi.org/