# A Framework for Secure Cloud-Empowered Mobile Biometrics

Aruna Sri Bommagani,* Matthew C. Valenti,* and Arun Ross†
*West Virginia University, Morgantown, WV, USA.
†Michigan State University, East Lansing, MI, USA.

***Abstract*—In this work, we describe how computationally intensive biometric recognition can be performed on a mobile device by offloading the actual recognition process to the cloud. We focus on facial recognition, though the paradigm can be applied to other modalities. We discuss a systematic approach for dividing a recognition operation and a bulk enrollment operation into multiple tasks, which can be executed in parallel on a set of servers in the cloud, and show how the results from each task can be combined and post-processed for individual recognition or template database generation. In the context of biometrics, preserving the privacy and security of biometric data is also of paramount interest. Therefore, we further explore the role of cancelable template generation for providing privacy protection when biometric data is stored in a cloud environment.**

## I. Introduction

In traditional identity management systems, user authentication is performed using passwords or ID cards. However, such systems have their disadvantages, as passwords and cards may be stolen, shared, or forgotten. Biometric recognition [1] offers an alternate solution to the user authentication problem as biometric traits cannot be easily lost, shared, or forgotten. A biometric system measures one or more physical or behavioral characteristics of an individual, such as fingerprint, face, or iris information, and attempts to automatically recognize the individual. The design of a biometric system includes *enrollment* and *recognition* phases. During the enrollment phase, biometric data is acquired from a user and stored in a database along with each subject's identity. During the recognition phase, biometric data is acquired and compared against the stored biometric data in order to establish the user's identity.

As biometric systems mature, two conflicting challenges have emerged. On the one hand, surges in enrollment and bulk matching operations can dramatically increase the computing requirements. On the other hand, the desire to implement biometric recognition on mobile, handheld systems will reduce the amount of local computing power available to the end users. These two challenges can be simultaneously adddressed by using cloud-computing resources, which allows computing to be performed remotely and treated as a utility [2]. However, it is not yet clear when and how to best leverage cloud computing for biometric applications. Furthermore, the risks of cloud-computing based biometric systems have not been fully characterized, and research needs to be directed towards mitigating these risks [3]. Paramount among these risks are security and privacy concerns [4], which are particularly acute when the biometric database is hosted by or transmitted to a public cloud service provider [5].

To date, the role of cloud computing within the context of biometric recognition systems has been considered in the literature from a variety of perspectives. A Hadoop-based [6] prototype for using the cloud for biometric identification is presented in [3]. However, it does not describe how to keep the biometric database secure. In [7], fingerprints are used to authenticate cloud users and cancelable biometrics are stored in the cloud, and [8] uses biometric identification to manage keys to access cryptographically encoded data stored on the cloud. While biometrics are an integral part of the security policies of [7] and [8], they are only used to authenticate the user and the matching is performed locally rather than in the cloud. In [9], erasures-coding is used to assure the integrity of data stored on the cloud and homomorphic tokens are used to detect intrusions. A privacy-preserving biometric identification scheme where the biometric database is encrypted and outsourced to the cloud servers is proposed in [10]. While [9] can detect a compromised database, and [10] provides a detailed security analysis to secure a biometric database, they offer no solution to minimize the damage resulting from a compromised biometric database. A conceptual design of secure mobile cloud platform using biometric encryption for mobile applications is proposed in [11], and secure authentication of mobile cloud users to protect cloud resources using a fingerprint image obtained using a mobile device camera is proposed in [12]. However, secure storage of templates and secret keys are not addressed in these works.

This paper investigates the use of cloud-computing technologies for performing biometric recognition and related tasks. Using facial recognition as an example, the paper considers the tradeoffs involved in architecting a system that can assure the privacy of the biometric database while realizing the computational advantages of cloud computing. To fully benefit from the massive parallelism offered by the cloud, a parallel and distributed algorithm for performing the biometric matching is developed and analyzed. To address the security concerns, a strategy for generating cancelable templates is presented. The concepts related to secure, distributed biometric recognition are embodied in a proof-of-concept mobile facial recognition system, whose architecture is fully described in this paper.

The remainder of this paper is organized as follows. Section II discusses key components in a typical facial recognition system and strategies for cancelable template generation. Sec-

tion III derives distributed algorithms for biometric template generation and parallel matching. Section IV describes the architecture of the proof-of-concept mobile biometric recognition system, while Section V provides an analysis of the system. A summary is presented in Section VI.

## II. Baseline Biometric System

In this work, we focus on facial recognition since face images can be easily acquired using a mobile device. Further, a number of applications can potentially benefit from performing face recognition efficiently in a mobile environment.

### A. LBP-Based Template Generation

A typical facial recognition system includes three main components: (1) face detection, (2) feature extraction, and (3) face matching. Face detection involves preprocessing of the image followed by a binary classifier to distinguish between a face and a non-face. In the present work, we perform face detection using the Viola-Jones detector [13]. Next, the face image is processed in order to extract features (template generation) that are necessary for face recognition. In the present work, we use Local Binary Patterns (LBP) [14], which is a texture-based face recognition technique. The final step in the recognition process is matching, where a probe or query is compared with gallery templates to generate matching scores and establish an identity. We have not used a commercial face matcher in this work since our approach requires direct access to the features extracted by the matcher, and commercial matchers do not offer this level of access.

The original form of LBP involves using pixels in a 3x3 pixel block and generating a label by comparing the values of each of the 8 pixels on the edge of the block with the value of the center pixel. For each of the 8 neighbors, if the value of the pixel is greater than the center pixel value, then a value of 1 is assigned, otherwise a value of 0 is assigned. The thresholding of the neighbors is performed in a circular fashion and the result is concatenated into a length-8 binary string, which is generally converted to its decimal value (label) between 0 and 255. Let $I_j$ be the $j^{th}$ grayscale face image, $j = \{1, ..., T\}$, where $T$ is the number of images in the database, and let $F_j$ be the corresponding LBP-labeled image. $I_j$ and $F_j$ are matrices of the same size. An *LBP histogram* $\mathbf{h}_j$ is generated for each $F_j$, which counts the number of occurrences of each distinct decimal label within $m$ *regions* of the image. $\mathbf{h}_j$ is stored as a column vector of length $\ell = mn$, where $m$ is the number of regions and $n$ is the number of distinct labels. The steps for computing $F_j$ and $\mathbf{h}_j$ are disclosed below.

A generalized form of LBP known as *multiscale LBP* allows for the use of a variable number of neighborhood pixels at different radii. The LBP algorithm with radius $R$ and $P$ neighbors in the circular neighborhood is denoted by $LBP_{P,R}$. A variant of multiscale LBP called *uniform LBP*, which is denoted by $LBP^{u_2}_{P,R}$, is used in this work. A label is said to be *uniform* if it has at most two bitwise transitions from 0 to 1 (or 1 to 0). In the corresponding histogram, each of the uniform patterns are assigned a separate label and all the non-uniform patterns are assigned to a single label. Therefore, the number of distinct output labels for mapping patterns with $P$ bits is $n = P(P-1) + 3$.

For a given grayscale image $I$, let $g_c$ denote the intensity of the $c^{th}$ pixel $(a_c, b_c)$ i.e., $g_c = I(a_c, b_c)$ and $g_p$ denote the the intensity of the $p^{th}$ neighbor of pixel $g_c$, $p = 0, ..., P-1$, where the neighbors are the $P$ pixels that are uniformly spaced on a circle of radius $R$ and centered at $(a_c, b_c)$. Bilinear interpolation is used to sample the image when the pixels do not lie entirely on the circle. The LBP label at pixel $(a_c, b_c)$ is given by

$$F(a_c, b_c) = \sum_{p=0}^{P-1} u[g_p - g_c] 2^p \tag{1}$$

where

$$u[z] = \begin{cases} 1, \text{if } z \geq 0 \\ 0, \text{otherwise.} \end{cases}$$

The image is divided into $m$ regions, $R_1, R_2, ...R_m$, and a histogram $\mathbf{h}^{(k)}$ is found for the labels of each $R_k$. The $i^{th}$ element of the LBP histogram for region $R_k, k = 1, ..., m$, is given by

$$h_i^{(k)} = \sum_{(a,b) \in R_k} \delta_i[F(a,b)] \tag{2}$$

for $i = 0, 1, ..., n-1$, where

$$\delta_i[z] \quad = \quad \begin{cases} 1, & z = i \\ 0, & z \neq i. \end{cases}$$

The LBP histograms for the $m$ regions are then stacked into a single column vector $\mathbf{h}$ of length $\ell = mn$. Thus, a biometric template is generated for each image $I_j$ by applying the above procedure, and in the case of $LBP^{u_2}_{P,R}$ the size of the template is $\ell = m\,(P(P-1) + 3)$.

### B. Matching

During the matching step, a probe image $\mathbf{x}$, which is itself represented as a LBP-labeled image, is compared against the gallery of templates, $\{\mathbf{h}_1, ...\mathbf{h}_T\}$, and the closest match is found. Here, we use minimum *Euclidian distance* as the selection criteria. The distance $d_j$ from the probe to each gallery template is computed according to:

$$d_j^2 \quad = \quad ||\mathbf{x} - \mathbf{h}_j|| = (\mathbf{x} - \mathbf{h}_j)^\top (\mathbf{x} - \mathbf{h}_j) \tag{3}$$

The closest image $\hat{I}_j$ is the one that minimizes the distance; i.e., the $I_j$ with index

$$j = \arg\min_j \{d_j\} \tag{4}$$

Once the most likely image $\hat{I}_j$ is determined, a final step is to identify the most likely subject. When there is a single image associated with each subject, this step is simply a matter of determining the subject associated with the image. In general, there are multiple images per subject, and the match will be on the subject associated with the most likely image. More generally, a ranked list of subjects can be provided to the user, listing the closest images in ascending order of distance.

### C. Cancelable Template Generation

In order to generate a cancelable template for $\mathbf{h}$, we first generate an $\ell \times \ell$ orthonormal matrix $A$. For additional security, an $\ell \times \ell$ secret permutation matrix $P$ and a length-$\ell$ blinding vector $\mathbf{b}$ can also be applied. Let the product of the orthonormal matrix $A$ and the random permutation matrix $P$ be the matrix $Q$. Then, the transformed template $\mathbf{y}$ of biometric template $\mathbf{h}$ can be given by

$$\mathbf{y} \quad = \quad (AP)\mathbf{h} + \mathbf{b} = Q\mathbf{h} + \mathbf{b}. \tag{5}$$

The orthonormal matrix $A$ can be generated using a secret key either by directly applying Gram-Schmidt orthogonalization [15] or by using the approach presented in [16], which is based on Gram-Schmidt orthogonalization. We follow the approach of [16], which begins by using a secret key to generate a set $\Theta = \{\theta_1, \theta_2, ..., \theta_v\}, \theta_i \in [0, 2\pi], v = \ell/2$, of random rotation angles. Define the $2 \times 2$ rotation matrix:

$$M(\theta) \quad = \quad \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{6}$$

The matrix $A$ is a block matrix composed by a $v \times v$ arrangement of $2 \times 2$ submatrices. More specifically, $A$ is block-diagonal whose diagonal entries are $M(\theta_1), ...M(\theta_v)$,

$$A \quad = \quad \begin{bmatrix} M(\theta_1) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & M(\theta_2) & \cdots & \mathbf{0} \\ \cdots & \cdots & & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & M(\theta_v) \end{bmatrix} \tag{7}$$

where each $\mathbf{0}$ in the above is a $2 \times 2$ matrix of zeros.

We can say that $\mathbf{y}$ is a *cancelable* template because if it is compromised, then the current transformed template can be revoked, and using a new key, i.e., a new set of $\Theta$, a new template can be regenerated. For an identification system, a single secret key can be used to generate cancelable templates for the entire gallery database.

During the matching stage, the distance from a transformed probe template is compared against the transformed gallery templates. Let $\mathbf{z} = AP\mathbf{x} + \mathbf{b}$ be the transformed probe template. The distance from $\mathbf{z}$ to each transformed gallery template, $\mathbf{y}_j$, is

$$d_j^2 \quad = \quad ||\mathbf{z} - \mathbf{y}_j|| = (\mathbf{z} - \mathbf{y}_j)^\top (\mathbf{z} - \mathbf{y}_j). \tag{8}$$

In order to maintain the same matching performance, it is essential that the distance profile be maintained. Thanks to the use of a unitary transformation matrix, this condition holds true. To see this, substitute (5) into (8),

$$\begin{aligned} ||\mathbf{z} - \mathbf{y}_j|| &= (\mathbf{z} - \mathbf{y}_j)^\top (\mathbf{z} - \mathbf{y}_j) \\ &= (Q\mathbf{x} + \mathbf{b} - Q\mathbf{h}_j - \mathbf{b})^\top (Q\mathbf{x} + \mathbf{b} - Q\mathbf{h}_j - \mathbf{b}) \\ &= (\mathbf{x} - \mathbf{h}_j)^\top Q^\top Q (\mathbf{x} - \mathbf{h}_j) \\ &= (\mathbf{x} - \mathbf{h}_j)^\top (\mathbf{x} - \mathbf{h}_j) = ||\mathbf{x} - \mathbf{h}_j|| \end{aligned} \tag{9}$$

where the last step follows from $Q^\top Q = I$. Thus the transformation used to generate the cancelable template does not change the distance profile, and will therefore leave the matching performance unchanged.

## III. Parallel computation

The two main phases in a biometric system are enrollment and recognition, which becomes increasingly computationally intensive as the size of the database grows. The effective turnaround time for operations in both phases can be greatly reduced through parallelization. In the context of a biometric system, parallelization involves breaking *jobs*, defined to be a user-requested action such as the enrollment of an individual or a recognition operation, into *tasks*, which are independent subsets of the operations required by a job, for instance a matching of a probe against just a subset of the biometric database. This section describes how to parallelize both the enrollment and recognition phases.

In general, a job involves the processing of $T$ templates. In the case of gallery generation or key renewal, $T$ images are transformed into $T$ cancelable templates. In the case of recognition, a single probe image is compared against the $T$ templates in the gallery. In each case, the job is easily parallelized by processing one or more templates per task. In the following discussion, we let $\eta$ be the number of tasks and $\lambda_i$ be the number of templates processed by the $i^{th}$ task. It follows that

$$T \quad = \quad \sum_{i=1}^{\eta} \lambda_i. \tag{10}$$

For ease of exposition, we assume that tasks are equally sized; i.e., $\lambda_i = \lambda, \forall i$, in which case $T = \eta\lambda$.

### A. Parallel biometric template generation

The complete gallery of templates can be generated in parallel by dividing the job of computing the $T$ templates into $\eta$ tasks, each involving the generation of $\lambda$ templates. Let $\mathcal{I}_k$ be the images associated with the $k^{th}$ task, and $\mathcal{Y}_k$ the corresponding cancelable templates. A job manager will create $\mathcal{I}_k$ by selecting a subset of the gallery containing $\lambda$ images. The package of images $\mathcal{I}_k$ will be sent to the appropriate computing asset, where it will be processed and the resulting $\mathcal{Y}_k$ computed. For each image $I \in \mathcal{I}_k$, the LBP label $F$ is generated according to (1), its histogram $\mathbf{h}$ computed according to (2), and a cancelable template $\mathbf{y}$ generated according to (5). On a parallel architecture, the $\{\mathcal{Y}_k\}$ can be processed in parallel, and then brought together at the server to form the complete gallery $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, ..., \mathcal{Y}_\eta\} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T\}$.

### B. Parallel distance matching

The distance between a transformed probe $\mathbf{z}$ and the gallery $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_T\}$ can be computed in parallel by dividing the job into $\eta$ tasks, each involving the matching of $\mathbf{z}$ against some subset of $\mathcal{Y}$. As with template generation, let $\mathcal{Y}_k$ represent the cancelable templates associated with the $k^{th}$ task. Note, however, that the size of the tasks used for template generation need not be the same as the size used for distance-based matching. In particular, distance calculation is much less complex than template generation, and therefore a distance-matching task will typically handle far more templates than will a template-generation task.

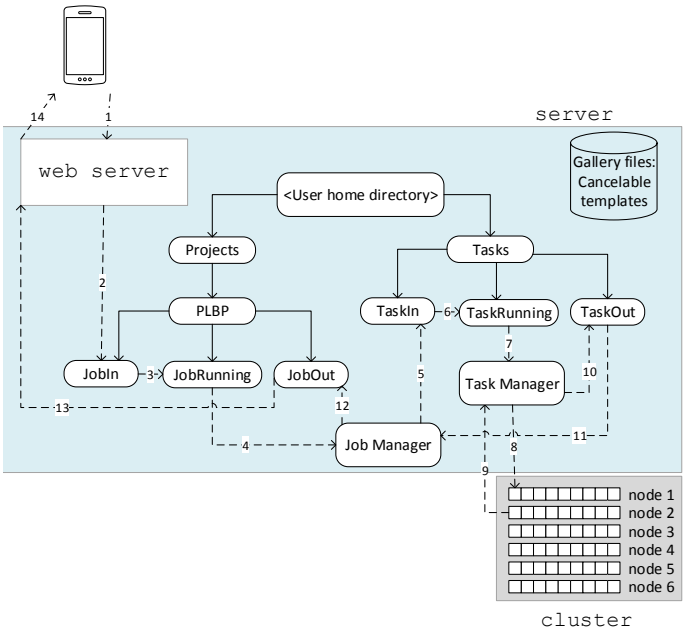


Fig. 1. System architecture and data flow.

For each task, the distance between $\mathbf{z}$ and each $\mathbf{y}_j \in \mathcal{Y}_k$ is found according to (8). The $\eta$ tasks are processed independently and in parallel, and the results returned to the server, which proceeds to identify the most likely match. On the one hand, the closest match can be found from the individual distances by picking the $I_j$ with index given by (4). Alternatively, the selection can be done in a hierarchical fashion, where a local most-likely candidate is found for each task, then the global most-likely candidate is found by picking the smallest distance among the $\eta$ local most-likely candidates.

## IV. Parallel LBP Facial Identification System

In this section, we present a complete mobile identification system that applies the approaches discussed in the previous sections to perform parallel template generation and matching on a computing cluster or cloud with a large number of processors. Because template generation involves accessing and modifying the model (gallery template data), only *privileged* users (e.g. an administrator) should be permitted to perform template generation. On the other hand, identification does not involve modification of data, and since the templates are cancelable, identification can be performed by a less-privileged *regular* user (e.g., an identification officer). More generally, the access privileges of users can be modified depending on the system policies.

By accessing the computing infrastructure through a mobile-friendly web interface, a regular user can take and upload an input image, view identification results (e.g., the three closest images to a probe image), and view results of previous identification jobs. A privileged user can furthermore enroll new subjects and generate cancelable face templates for either the entire image gallery (which is required during key renewal) or for a newly enrolled subject.

### A. Components of the System

The main components of the system are the mobile devices (e.g. cell phones or tablet computers), a system server running a web server, a system server, and a job manager, and the computational assets running on the cluster. The system architecture, its components and the flow of data is shown in Fig. 1. Each *mobile device* includes a camera for taking the probe image, a communications interface (e.g., WiFi or LTE cellular) for uploading the image, and a mobile web browser for viewing results. The *web server* hosts the web application that can be accessed using the mobile devices. Job requests are submitted and retrieved by the mobile user via the web interface. The *computing cluster* includes a set of networked computers, which together process the user-submitted jobs by operating as an integrated computing resource. The *job manager* is a process that divides jobs into tasks, manages the execution of jobs, and monitors user activity and usage. The *task manager* is a process that coordinates parallel execution of tasks on the cluster. Tasks are serviced by *workers*, which are agents running on the computing cluster that will execute the individual tasks.

As mentioned above, a *job manager* is a process that divides each job into tasks, tracks the job status by monitoring the execution of tasks, post-processes the completed tasks (for instance, by sorting the distances computed by the tasks in an effort to determine the most-likely subject), and maintains a record of the computational effort required for each task. The job manager furthermore keeps track of each user by monitoring the number of jobs that the user has submitted, and the amount of computing resources consumed by each job and user of the system. Usage quotas can be imposed to prevent a single user from consuming too much of the computing resource, and a biometrics-as-a-service system can bill based on the amount of usage.

Each job and task in the system is embodied by a data file in the filesystem containing the values required to execute the job. While the architecture is agnostic to the programming language, our implementation is based on Matlab, and the data files are stored in the .mat file format. Within the filesystem, a *queue* is a directory holding a file that embodies a job or a task. The job and task manager each maintain its own sets of queues, and each manager has access to three kinds of queues: Input, Running, and Output. Multiple algorithms and modalities are supported through the use of different *projects*, where a *project* is a directory in the filesystem that holds the job queues of a specific recognition algorithm (and the procedure to generate cancelable templates). Different recognition algorithms or biometric modalities can be implemented as different projects.

A recognition job is submitted by uploading an image through the mobile web interface. The web server will create the appropriate job file containing the necessary algorithmic parameters and will place it into the `job input` queue for that user. As each user will have its own job queue, the *job manager* will sweep through the job input queues of all the users and will select a job for execution using a scheduling

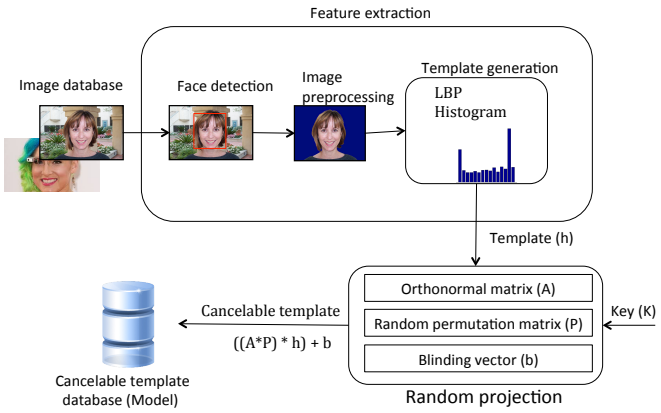


Fig. 2. Parallel enrollment.

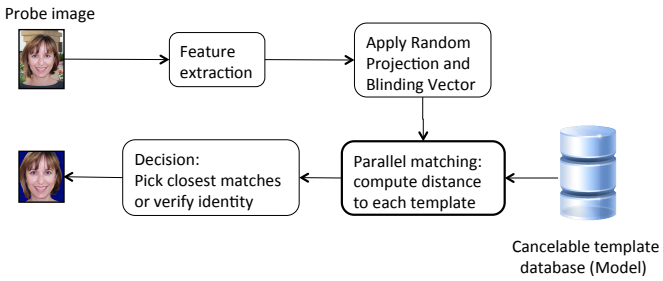


Fig. 3. Parallel face recognition.

policy such as first-in first-out (FIFO), fair, or proportionally fair. The job will be divided into $\eta$ tasks, and each task placed into the user's `task input` queue.

If the time required to process the $k^{th}$ task is $t_k$, which may vary depending on parameters such as the processing power of the worker, then the *total processor time* required to complete a job, $\Gamma_p = \sum_{k=1}^{\eta} t_k$, where $\eta$ is the number of tasks required to complete a job. The *duration* of a job, $\Gamma_c$, is the difference between the clock time that a job is completed and when it was submitted. On a uniprocessor machine $\Gamma_c = \Gamma_p$, but on a massive parallel architecture $\Gamma_c << \Gamma_p$ since many tasks may be run in parallel.

### B. Job preprocessing

When the job manager selects a job for execution, the corresponding job file is moved from the `job input` queue to the `job running` queue. Concurrently, the job manager performs preprocessing of the job and splits it into $\eta$ tasks, each of which is embodied by a task file placed into the `task input` queue. In the case of parallel template generation, preprocessing includes the validation of key and gallery data. Depending on the number of enrolled subjects (and images per subject), an optimal number of tasks is evaluated (dependent on the number and type of resources available for parallel execution). For an identification job, preprocessing includes gathering volume information of gallery template database (model) to calculate the number of tasks required to establish identity of a probe image.

The *key* is an important parameter when using cancelable template generation, and it should be noted that cancelable template generation is invertible; i.e., if an adversary has access to key and algorithmic details to generate cancelable templates, then the original biometric template can be compromised. To support the validity of the approach in Section II-C, robustness and diversity properties are discussed in [16]. In addition to this, since the security, integrity, and confidentiality of machines on a distributed environment is questionable, care is taken such that a hash of a key is stored on a secure system server using Bcrypt [17], and only the information required for transformation such as matrix $Q$ and vector $\mathbf{b}$ are included in a task.

### C. Job execution

After preprocessing, based on the number of tasks required to complete execution of a job, the job manager generates and queues tasks in *task input*. Each task is generated such that it includes all the required information for its execution without any dependency on other tasks or on the job manager, thus, preventing data corruption and race conditions.

A template generation task may include parameters such as the location of the directory hosting the image gallery, algorithm-specific information such as the radius $R$ and the number of neighbors within the radius $P$, and parameters for generating the cancelable template such as the permuted projection matrix $Q$ and blinding vector $\mathbf{b}$. In the case of a matching task, a link to the probe $\mathbf{z}$ must be provided in the task along with the location of the gallery of secure template.

Each task in `task input` queue is randomly picked up by a worker and moved to `task running` queue when the worker begins executing the task. Parallel template generation can be seen in Fig. 2, as described in Section III-A. Parallel matching is performed as described in Section III-B and Fig. 3. After task execution, a worker moves a task to `task output` queue. Job manager picks and consolidates results from all the completed tasks. After the final task execution of a job, the job manager updates job results (model or identity), total duration ($\Gamma_c$), and total processor time ($\Gamma_p$), and moves the job to `job output` queue with all the necessary output data that can be viewed using the mobile web interface.

## V. Identification System Analysis

In order to evaluate the system, we ran a test on a cluster computer comprising 22 servers containing a total of 396 cores for running workers. The database used is XM2VTSDB [18], which is designed to support multi-modal biometric research. It contains face data of 295 subjects recorded on a uniform blue background taken in controlled lighting conditions. For each of the 295 identities, a total of eight images were acquired, and in our evaluation, we use four images per subject as gallery images and the remaining four images are used as probes.

$LBP^{u2}_{P,R}$ has two important parameters: the radius, $R$, and the number of neighbors, $P$, within $R$. We ran simulations using different values for $R$ and $P$ in order to determine optimum parameter values. Fig. 4 represents identification accuracy rate as a function of number of neighbors, $P$, and Fig. 5 represents identification accuracy rate as a function of

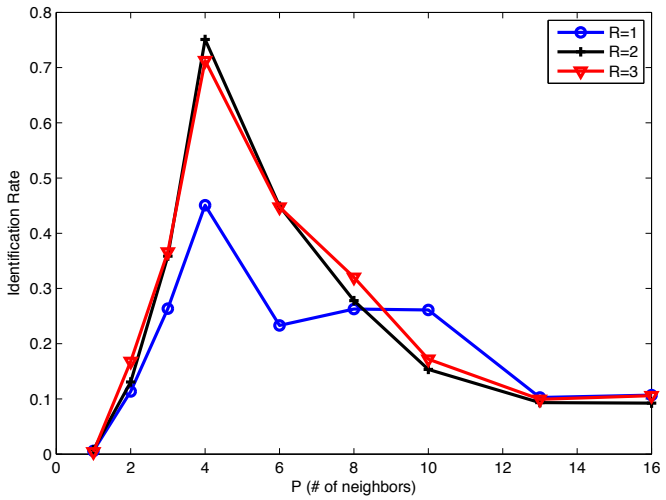


Fig. 4. Identification rate as a function of number of neighbors using LBP algorithm parameterized by different radii.

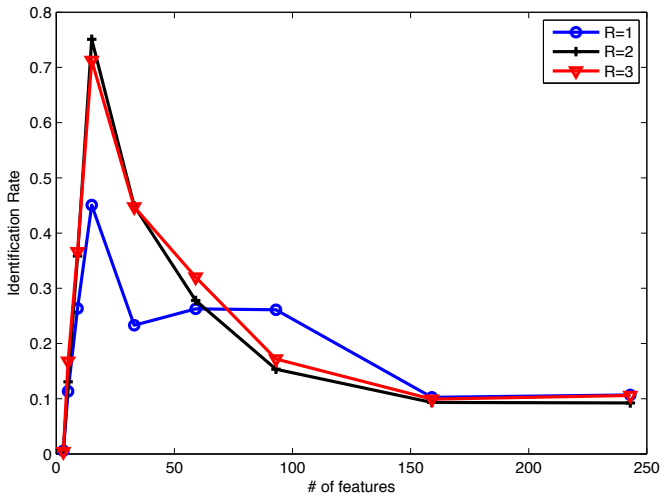


Fig. 5. Identification rate as a function of the number of features using LBP algorithm parameterized by different radii.

number of features in an image template. From the figures, we can observe that highest identification rate of 75.08% is obtained with $P = 4$ and $R = 2$. We can also observe that for a particular radius in Fig. 4, the identification rate increases and reaches its optimum at $P = 4$, and then starts decreasing; there is only a slight variation in accuracy rate when $R = 2$ and $R = 3$. As the template size (i.e., number of features) is a function of $P$, we observe a similar phenomenon for identification rate when it is plotted as a function of number of features as shown in Fig. 5.

In Section II-C, we emphasized that if the same key is used for transformation of templates, then the distance between them is preserved before and after the transformation. Fig. 6 is a set of cumulative match characteristic (CMC) curves obtained for $P = 4$ and $R = 2, 3$ for original and cancelable templates. We can observe that securing templates using transformation indeed preserves identification accuracy. Also, within the top 10 ranks the identity of a test image can be correctly established with an accuracy rate of 93.47%. Table I

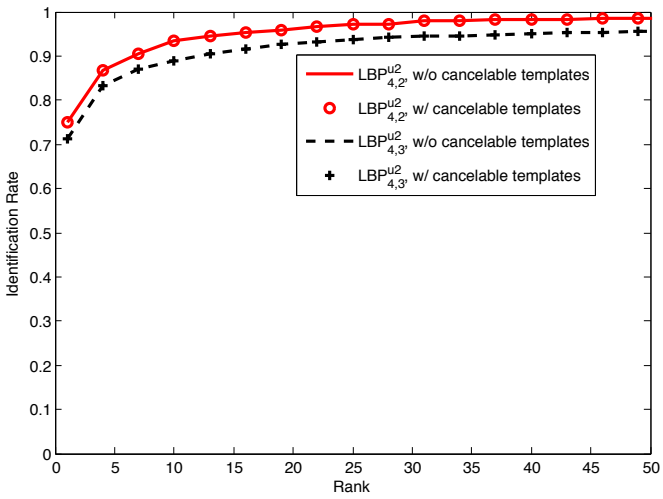


Fig. 6. CMC curves using LBP with uniform mapping for original and cancelable templates for P = 4 and R = 2, 3.

TABLE I
IDENTIFICATION RATE CORRESPONDING TO RANK USING LBP FOR P = 4 AND R = 2, 3.

| Rank | $LBP^{u2}_{4,2}$ | $LBP^{u2}_{4,3}$ |
|---|---|---|
| 1 | 0.7508 | 0.7127 |
| 2 | 0.8220 | 0.7797 |
| 4 | 0.8669 | 0.8331 |
| 6 | 0.8915 | 0.8576 |
| 8 | 0.9110 | 0.8754 |
| 10 | 0.9347 | 0.8890 |

shows the identification accuracy rate corresponding to a rank for P = 4 and R = 2, 3. Fig. 7 shows the receiver operating characteristic (ROC) curves obtained by using uniform LBP with $P = 4$ and $R = \{2, 3\}$.

The difference in computational performance when using a full cluster to a set of different types of nodes can be observed in Fig. 8. The figure shows the number of comparisons (between a probe and a gallery template) performed as a function of time (in seconds), and Table II provides average number of comparisons performed per second. Nodes in a node type are categorized based on the computational capability. From Table II, we can observe 10 to 30 times improvement in throughput when the full cluster is utilized to process jobs.

## VI. CONCLUSION

This paper has presented a framework to perform biometric face recognition in a cloud environment where the enrollment and identification operations are divided into multiple tasks, which can be run in parallel while maintaining the security of the biometric templates. By using cancelable template protection, biometric templates are made secure, which is important when the container holding the templates are not completely secure, such as in a public cloud setting. The cancelable templates can be revoked and regenerated whenever required or according to proactive security policy.

In addition, we have shown how to adapt the LBP face recognition paradigm to run on a cloud or cluster computing

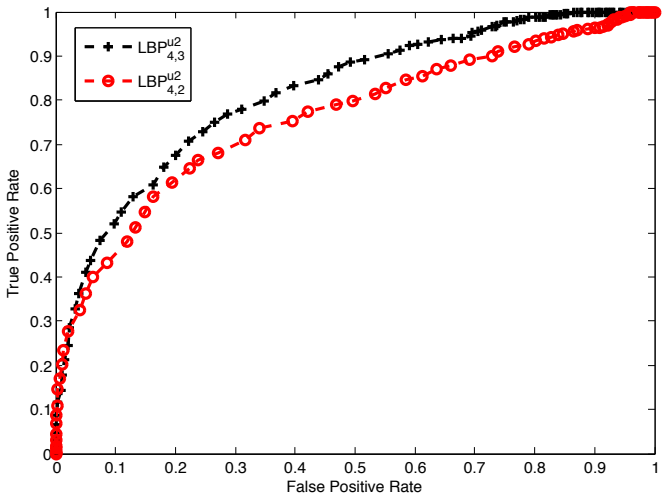


Fig. 7. ROC curves using uniform-LBP for P = 4 and R = 2, 3.

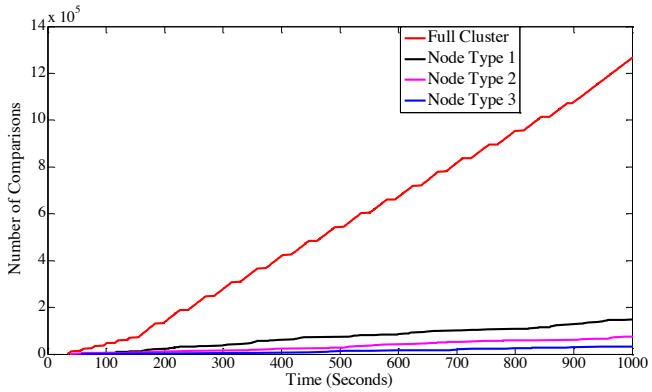


Fig. 8. Computational performance.

environment by describing how to parallelize the template-generation and face-matching algorithms, and showing the key components of the architecture. An analysis of the system where tasks are run in parallel on a 396-core cluster using XM2VTS face database is provided. A 10- to 30-fold improvement in throughput is observed. However, for the XM2VTS database, there is negligible decrease in latency when compared against a stand-alone system, and we anticipate that the computational benefits of parallel computing will be more dramatic when a larger database or more computationally intensive matching algorithm is used.

The proposed work investigated the feasibility of secure cloud-empowered mobile biometric identification systems, and the metrics developed can be used as benchmarks for future cloud computing based biometric systems. The possible improvements to the proposed work include incorporating improved face recognition algorithms and applying the proposed approaches to use other modalities or multimodal biometrics. We would also like to research and formulate improved key management and access policies, and address scalability issues.

## ACKNOWLEDGMENT

This research was funded by the Center for Identification Technology Research (CITeR), a National Science Foundation (NSF) Industry/University Cooperative Research Center (I/UCRC).

TABLE II
COMPUTATIONAL PERFORMANCE: AVERAGE NUMBER OF TEMPLATE COMPARISONS PER SECOND

| Computational entity type | Number of comparisons |
|---|---|
| Full cluster | 1348.4 |
| Node type 1 | 127.49 |
| Node type 2 | 80.74 |
| Node type 3 | 44.75 |

## REFERENCES

[1] A. K. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, pp. 4–20, Jan. 2004.

[2] R. Das, "Biometrics in the cloud," *Keesing Journal of Documents and Identity*, pp. 21–23, Feb. 2013.

[3] E. Kohlwey, A. Sussman, J. Trost, and A. Maurer, "Leveraging the cloud for big data biometrics: Meeting the performance requirements of the next generation biometric systems," in *Proc. IEEE World Congress on Services*, (Los Alamitos, CA, USA), pp. 597–601, Jul. 2011.

[4] S. Prabhakar, S. Pankanti, and A. K. Jain, "Biometric recognition: Security and privacy concerns," *IEEE Security & Privacy*, vol. 1, pp. 33–42, Mar. 2003.

[5] X. Zhifeng and X. Yang, "Security and privacy in cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 843–859, 2013.

[6] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, third ed., May. 2012.

[7] J. Yang, N. Xiong, A. V. Vasilakos, Z. Fang, D. Park, X. Xu, S. Yoon, S. Xie, and Y. Yang, "A fingerprint recognition scheme based on assembling invariant moments for cloud computing communications," *IEEE Systems Journal*, vol. 5, pp. 574–583, Dec. 2011.

[8] D. G. Martínez, F. J. G. Castaño, E. A. Rúa, J. L. A. Castro, and D. A. R. Silva, "Secure crypto-biometric system for cloud computing," in *Proc. 1$^{st}$ Int'l Workshop on Securing Services on the Cloud*, pp. 38–45, Sep. 2011.

[9] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. 17$^{th}$ Int'l Workshop on Quality of Service*, pp. 1–9, Jul. 2009.

[10] J. Yuan and S. Yu, "Efficient privacy-preserving biometric identification in cloud computing," in *Proc. IEEE INFOCOM*, pp. 2652–2660, Apr. 2013.

[11] K. Zhao, H. Jin, D. Zou, G. Chen, and W. Dai, "Feasibility of deploying biometric encryption in mobile cloud computing," in *Proc. 8$^{th}$ ChinaGrid Annual Conf.*, pp. 28–33, Aug 2013.

[12] I. A. Rassan and H. AlShaher, "Securing mobile cloud using finger print authentication," *International Journal of Network Security & Its Applications*, vol. 5, Nov. 2013.

[13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, 2001.

[14] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 2037–2041, Dec. 2006.

[15] G. B. Arfken, "Gram-schmidt orthogonalization," in *Mathematical Methods for Physicists*, pp. 516–520, Orlando, FL: Academic Press, 3rd ed., 1985.

[16] H. Al-Assam, H. Sellahewa, and S. Jassim, "A lightweight approach for biometric template protection," in *Proc. SPIE, Mobile Multimedia/Image Processing, Security, and Applications*, vol. 7351, May. 2009.

[17] N. Provos and D. Mazières, "A future-adaptable password scheme," in *Proc. USENIX Annual Technical Conf.*, (Monterey, California, USA), pp. 81–92, Jun. 1999.

[18] K. Messer, J. Matas, J. Kittler, J. Lüttin, and G. Maitre, "XM2VTSDB: the extended M2VTS database," in *Proc. Second Int'l Conf. on Audio and Video-based Biometric Person Authentication*, pp. 72–77, 1999.