Blind Authentication: A Secure Crypto-Biometric Verification Protocol

Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan and C.V. Jawahar
{upmanyu@research., anoop@, srinathan@, jawahar@}@iiit.ac.in
International Institute of Information Technology, Hyderabad, INDIA - 500032

***Abstract*—Concerns on widespread use of biometric authentication systems are primarily centered around template security, revocability and privacy. The use of cryptographic primitives to bolster the authentication process can alleviate some of these concerns as shown by biometric cryptosystems. In this paper, we propose a *provably secure* and *blind* biometric authentication protocol, which addresses the concerns of user's privacy, template protection, and trust issues. The protocol is blind in the sense that it reveals only the identity, and no additional information about the user or the biometric to the authenticating server or vice-versa. As the protocol is based on asymmetric encryption of the biometric data, it captures the advantages of biometric authentication as well as the security of public key cryptography. The authentication protocol can run over public networks and provide non-repudiable identity verification. The encryption also provides template protection, the ability to revoke enrolled templates, and alleviates the concerns on privacy in widespread use of biometrics.**

The proposed approach makes no restrictive assumptions on the biometric data and is hence applicable to multiple biometrics. Such a protocol has significant advantages over existing biometric cryptosystems, which use a biometric to secure a secret key, which in turn is used for authentication. We analyze the security of the protocol under various attack scenarios. Experimental results on four biometric datasets (face, iris, hand geometry and fingerprint) show that carrying out the authentication in the encrypted domain does not affect the accuracy, while the encryption key acts as an additional layer of security.

***Index Terms*—Biometrics, Privacy, Security, Cryptosystems, Support Vector Machines, Artificial Neural Networks, Public Key Cryptography.** [1]

EDICS Category: MOD-SECU, BIO-PROT, BIO-ATTA, SEC-PRIV

I. Introduction

BIOMETRIC authentication systems are gaining widespread popularity in recent years due to the advances in sensor technologies as well as improvements in the matching algorithms [1] that make the systems both secure and cost-effective. They are ideally suited for both high security and remote authentication applications due to the non-repudiable nature and user convenience. Most biometric systems assume that the template in the system is secure due to human supervision (e.g., immigration checks and criminal database search) or physical protection (e.g., laptop locks and door locks). However, a variety of applications of authentication need to work over a partially secure or insecure networks such as an ATM networks or the Internet. Authentication over insecure public networks or with untrusted servers raises more concerns in privacy and security. The primary concern is related to the security of the plain biometric templates, which cannot be replaced, once they are compromised [2]. The privacy concerns arise from the fact that the biometric samples reveal more information about its owner (medical, food habits, etc.) in addition to the identity. Widespread use of biometric authentication also raises concerns of tracking a person, as every activity that requires authentication can be uniquely assigned to an individual (see Table I).

To clarify our problem let us consider the following usage scenario: *"Alice wants to create an account in Bobmail, that requires biometrics based authentication. However, she neither trusts Bob to handle her biometric data securely, nor trusts the network to send her plain biometric."*

The primary problem here is that, for Alice, Bob could either be incompetent to secure her biometric or even curious to try and gain access to her biometric data, while the authentication is going on. So Alice does not want to give her biometric data in plain to Bob. On the other hand, Bob does not trust the client as she could be an impostor. She could also repudiate her access to the service at a later time. For both parties, the network is insecure. A biometric system that can work securely and reliably under such circumstances can have a multitude of applications varying from accessing remote servers to e-shopping over the Internet. Table I summarizes the primary concerns that needs to be addressed for widespread adoption of biometrics. For civilian applications, these concerns are often more serious than the accuracy of the biometric [3].

If the user is able to authenticate himself using a strongly encrypted version of his biometric (say using RSA [4]), then many of the concerns on privacy and security can be addressed. However, this would require the server to carry out all the computations in the encrypted domain itself. Unfortunately, encryption algorithms are designed to remove any similarity that exist within the data to defeat attacks, while pattern classification algorithms require the similarity of data to be preserved to achieve high accuracy. In other words, security/privacy and accuracy seems to be opposing objectives. Different secure authentication solutions try to make reasonable trade-offs between the opposing goals of security and accuracy, in addition to making specific assumptions about the representation or biometric being used.

[1]Copyright (c) 2008 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org

a) Template protection: As a biometric do not change over time, one cannot revoke an enrolled plain biometric. Hence, critical information could be revealed if the server's biometric template database is compromised.
b) User's privacy: i) The activities of a person could be tracked, as the biometric is unique to a person, and ii) Certain biometrics may reveal personal information about a user (e.g., medical or food habits), in addition to identity.
c) Trust between user and server: In widespread use, all authenticating servers may not be competent or trustworthy to securely handle a user's plain biometric, while a remote user cannot be reliably identified without biometric information.
d) Network security: As the authentication is done over an insecure network, anyone snooping the network could gain access to the biometric information being transmitted.

TABLE I
PRIMARY CONCERNS IN WIDESPREAD ADOPTION OF BIOMETRICS FOR REMOTE AUTHENTICATION.

We overcome this seemingly unavoidable compromise by designing the classifier in the plain feature space, which allows us to maintain the performance of the biometric. We would then like to carry out the computations required for authentication using this trained classifier, completely in the encrypted domain. However, such a solution would require an *algebraic homomorphic encryption* scheme [5]. The only known doubly homomorphic scheme has recently been proposed by Craig Gentry [6] and would mostly lead to a computationally intensive theoretical solution. We show that it is possible to achieve a practical solution using distribution of work between the client (sensor) and the server (authenticator), using our proposed randomization scheme.

A. Previous Work

The previous work in the area of encryption based security of biometric templates tend to model the problem as that of building a classification system that separates the genuine and impostor samples in the encrypted domain [7] [8] [9]. However a strong encryption mechanism destroys any pattern in the data, which adversely affects the accuracy of verification. Hence, any such matching mechanism necessarily makes a compromise between template security (strong encryption) and accuracy (retaining patterns in the data). The primary difference in our approach is that we are able to design the classifier in the plain feature space, which allows us to maintain the performance of the biometric itself, while carrying out the authentication on data with strong encryption, which provides high security/privacy [10].

Over the years a number of attempts have been made to address the problem of template protection and privacy concerns and despite all efforts, as A.K. Jain *et al.* puts it, *a template protection scheme with provable security and acceptable recognition performance has thus far remained elusive.* [9]. In this section, we will look at the existing work in light of this security-accuracy dilemma, and understand how this can be overcome by communication between the authenticating server and the client. Detailed reviews of the work on template protection can be found in Jain *et al.* [9], Uludag *et al.* [11], and Ratha *et al.* [12]. We will adopt the classification of existing works provided by Jain *et al.* [9] (see Fig 1), and show that each class of approaches makes the security-accuracy compromise.

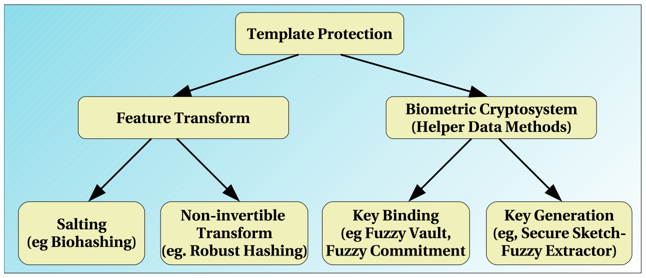

Template Protection
Feature Transform
Biometric Cryptosystem (Helper Data Methods)
Salting (eg Biohashing)
Non-invertible Transform (eg. Robust Hashing)
Key Binding (eg Fuzzy Vault, Fuzzy Commitment
Key Generation (eg, Secure Sketch-Fuzzy Extractor)

Fig. 1. Categorization of template protection schemes by Jain *et al.* [9].

Let us now analyze each of the four category of solutions in terms of their strengths and weaknesses:

The first class of feature transformation approaches known as *Salting* offers security using a transformation function seeded by a user specific key. The strength of the approach lies in the strength of the key. A classifier is then designed in the encrypted feature space. Although the standard cryptographic encryption such as AES or RSA offers secure transformation functions, they cannot be used in this case. The inherent property of dissimilarity between two instances of the biometric trait from the same person, leads to large differences in their encrypted versions. This leads to a restriction on the possible functions that can be used and in salting, resulting in a compromise made between security and the performance. Some of the popular salting based approaches are biohashing [13] [8] and salting for face template protection [14]. Moreover, salting based solutions are usually specific to a biometric trait, and in general do not offer well defined security. Kong *et al.* do a detailed analysis of the current biohashing based biometric approaches [15]. They conclude that the zero EER reported by many papers is obtained in carefully set experimental conditions and unrealistic under assumptions from a practical view point.

The second category of approaches identified as *Non-invertible transform* applies a trait specific non-invertible function on the biometric template so as to secure it. The parameters of the transformation function are defined by a key which must be available at the time of authentication to transform the query feature set. Some of the popular approaches that fall into this category are Robust Hashing and Cancelable Templates. Cancelable templates [12], [16] allows one to replace a leaked template, while reducing the amount of information revealed through the leak, thus addressing some of the privacy concerns. However, such methods are often biometric specific and do not make any guarantees on preservation of privacy [17], especially when the server is not trusted. Methods to detect tampering of the enrolled templates [18] help in improving the security of the overall system.

Boult *et al.* [17] extended the above approach to stronger encryption, and proposed an encrypted minutia representation and matching scheme of fingerprints. The position information of a minutia is divided into a stable integer part and a variable increment. A *Biotoken* consists of the encrypted integer part and the increment information in plain. A specific matching algorithm was proposed to match the biotokens for verification. The approach provides provable template security as a strong encryption is used. Moreover, the matching is efficient, and is shown to even improve the matching accuracy. However, the primary fact that encryption is applied to part of the data, which itself is quantized, may mean some amount of compromise between security and accuracy. An extension to the above work based on re-encoding methodology for revocable biotokens is proposed by the authors in [19]. In this method, the computed biotoken is re-encoded using a series of unique new transformation functions to generate a *Bipartite Biotoken*. For every authentication, the server computes a new bipartite biotoken, which is to be matched by the client against the biotoken generated by him. The method significantly enhances the template security as compared to the original protocol. Moreover, as bipartite biotoken is different for each authentication request, replay attacks are not possible. However, in the current form, the base biotoken is available (in plain) with the server, and if the biotoken database is compromised, a hacker can gain access to all the users' accounts until the biotokens are replaced. The method aims at securing the actual biometric template, which cannot be recovered from a secure biotoken.

The third and fourth classes, shown in Fig 1, are both variations of *Biometric cryptosystems*. They try to integrate the advantages of both biometrics and cryptography to enhance the overall security and privacy of an authentication system. Such systems are primarily aimed at using the biometric as a protection for a secret key (Key Binding approach [20]) or use the biometric data to directly generate a secret key (Key Generation approach [21]). The authentication is done using the key, which is unlocked/generated by the biometric. Such systems can operate in two modes in the case of remote authentication. In the first case, the key is unlocked/generated at the client end, which is sent to the server for authentication, which will ensure security of the template, and provide user privacy. However, this would become a key based authentication scheme and would lose the primary advantage of biometric authentication, which is its non-repudiable nature. In the second case, the plain biometric needs to be transmitted from the user to the server, both during enrollment and during authentication. This inherently leaks more information about the user than just the identity, and the users need to trust the server to maintain their privacy (concerns Table I: *b* and *c*). Moreover, authenticating over an insecure network makes the plain biometric vulnerable to spoofing attacks (concerns Table I: *d*).

Biometric cryptosystem based approaches such as Fuzzy Vault and Fuzzy extractor in their true form lack diversity and revocability. According to Jain *et al.* [9], a performance degradation usually takes place as the matching is done using error correction schemes. This precludes the use of sophisticated matchers developed specifically for matching the original biometric template. Biometric cryptosystems, along with salting based approaches introduce diversity and revocability in them. Moreover, Walter *et al.* [22] demonstrated a method for recovering the plain biometric from two or more independent secrets secured using the same biometric. A detailed review of the previous work in this area can be found in Uludag et al. [11] and Jain et al [9].

Nagai *et al.* [23] proposed the use of client side computation for part of the verification function. Their approach, termed *ZeroBio*, models the verification problem as classification of a biometric feature vector using a 3-layer neural network. The client computes the outputs of the hidden layer, which is transferred to the server. The client then proves to the server that the computation was carried out correctly, using the method of zero-knowledge proofs. The server completes the authentication by computing the output values of the neural network. The method is both efficient and generic as it only requires computation of weighted sums and does not make any assumption on the biometric used. It also provides provable privacy to the user, as the original biometric is never revealed to the server. However, the system requires that the hidden layer weights be transferred to the server without encryption. This allows the server to estimate the weights at the hidden layer from multiple observations over authentications. Once the weights are known, the server can also compute the feature vector of the biometric, thus compromising both security and privacy. The system could also be compromised if an attacker gains access to the client computer, where the weight information is available in plain.

Blind authentication, proposed in our paper, is able to achieve both strong encryption based security as well as accuracy of a powerful classifiers such as support vector machines (SVM [24]) and Neural Networks [25]. While the proposed approach has similarities to the Blind Vision [26] scheme for image retrieval, it is far more efficient for the verification task.

Blind Authentication addresses all the concerns mentioned in Table I.

1) The ability to use strong encryption addresses template protection issues as well as privacy concerns.
2) Non-repudiable authentication can be carried out even between non-trusting client and server using a trusted third party solution.
3) It provides provable protection against replay and client-side attacks even if the keys of the user are compromised.
4) As the enrolled templates are encrypted using a key, one can replace any compromised template, providing revocability, while allaying concerns of being tracked.

In addition, the framework is generic in the sense that it can classify any feature vector, making it applicable to multiple biometrics. Moreover, as the authentication process requires someone to send an encrypted version of the biometric, the non-repudiable nature of the authentication is fully preserved, assuming that spoof attacks are prevented. Note that the proposed approach does not fall into any of the categories given in figure 1. This work opens a new direction of research to look at privacy preserving biometric authentication.

II. Blind Authentication

We define *Blind Authentication* as "A biometric authentication protocol that does not reveal any information about the biometric samples to the authenticating server. It also does not reveal any information regarding the classifier, employed by the server, to the user or client". Note that such a protocol can satisfy the conditions presented in our initial scenario, where Alice wanted to create an account with Bobmail that required biometric authentication, whom she did not trust. We now present the authentication framework that achieves this goal using any biometric, and prove that the information exchanged between the client and the server does not reveal anything other than the identity of the client.

For the sake of simplicity, we initially assume that authentication is done through a generic *linear classifier*. We later describe, how the protocol can be extended to more generic and powerful classifiers, like the *Support Vector Machine* (SVM [24]) and the *Neural Networks* [27] [25]. One could use any biometric in this framework as long as each test sample is represented using a feature vector x of length n. Note that even for biometrics such as fingerprints, one can define fixed length feature representations [7].

Let ω be the parameters of the linear classifier (perceptron). The server accepts the claimed identity of a user, if $\omega \cdot x < \tau$, where τ is a threshold. As we do not want to reveal the template feature vector (ω) or the test sample (x) to the server, we need to carry out the perceptron function computation directly in the encrypted domain. Computing $\omega \cdot x$ involves both multiplication and addition operations, thus computing it in the encrypted domain requires the usage of a doubly homomorphic encryption scheme [28]. In the absence of a practical doubly homomorphic encryption scheme (both additive and multiplicative homomorphic), our protocol uses a class of encryption that are multiplicative homomorphic, and we simulate addition using a clever randomization scheme over one-round of interaction between the server and the client. An encryption scheme, $E(x)$ is said to be multiplicative homomorphic, if $E(x)E(y) = E(xy)$ for any two numbers x and y. We use the popular RSA encryption scheme [4], which satisfies this property.

An overview of the authentication process is presented in Fig 2. We assume that the server has the parameter vector ω in the encrypted form, i.e., $E(\omega)$, which it receives during the enrollment phase. The authentication happens over two rounds of communication between the client and the server.

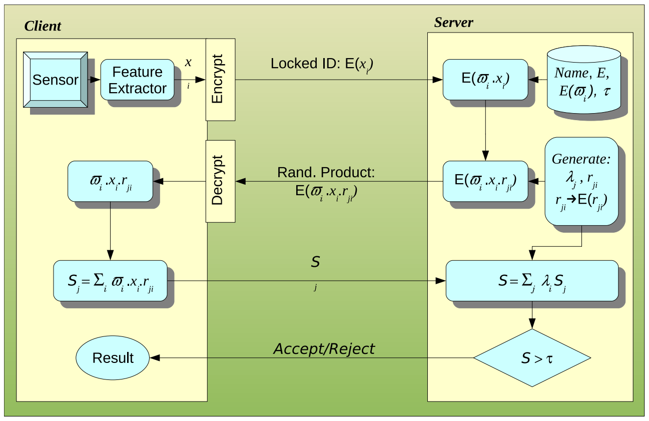

Client
Server
Sensor
Feature Extractor
x_i
Encrypt
Locked ID: E(x_i)
E($\varpi_i . x_i$)
Name, E, E(ϖ_i), τ
Decrypt
Rand. Product: E($\varpi_i . x_i . r_{ji}$)
$\varpi_i . x_i . r_{ji}$
E($\varpi_i . x_i . r_{ji}$)
Generate: λ_j, r_{ji} $r_{ji} \rightarrow$ E(r_{ji})
$S_j = \Sigma_i \varpi_i . x_i . r_{ji}$
S_j
$S = \Sigma_j \lambda_i S_j$
Result
Accept/Reject
$S > \tau$

Fig. 2. Blind Authentication Process: Linear kernel computation for encrypted feature vectors. At no point, the identity vectors x, ω or the intermediate results $x_i \cdot \omega_i$ is revealed to anyone.

To perform authentication, the client locks the biometric test sample using her public key and sends the *locked ID* to the server. The server computes the products of the locked ID with the locked classifier parameters and randomizes the results. These *randomized products* are sent back to the client. During the second round, the client unlocks the randomized results and computes the sum of the products. The resulting *randomized sum* is sent to the server. The server de-randomizes the sum to obtain the final result, which is compared with a threshold for authentication.

As we described before, both the user (or client) and the server do not trust each other with the biometric and the claimed identity. While the enrollment is done by a trusted third party, the authentications can be done between the client and the server directly. The client has a biometric sensor and some amount of computing power. The client also possesses an RSA private-public key pair, E and D. We will now describe the authentication and enrollment protocols in detail.

A. Authentication

We note that the computation of: $\omega \cdot x$ requires a set of scalar multiplications, followed by a set of additions. As the encryption used (RSA) is homomorphic to multiplication, we can compute, $E(\omega_i x_i) = E(\omega_i)E(x_i)$, at the server side. However, we cannot add the results to compute the authentication function. Unfortunately, sending the products to the client for addition will reveal the classifier parameters to the user, which is not desirable. We use a clever randomization mechanism that achieves this computation without revealing any information to the user. The randomization makes sure that the client can do the summation, while not being able to decipher any information from the products. The randomization is done in such a way that the server can compute the final sum to be compared with the threshold. The overall algorithm of the authentication process is given in Algorithm 1. Note that all the arithmetic operations that we mention in the encrypted domain will be $modulo-$ operations, i.e. all the computations such as (a *op* b) will be done as (a *op* b) mod p, where p is defined by the encryption scheme employed.

In the algorithm, the server carries out all its computation in the encrypted domain, and hence does not get any information about the biometric data (x) or the classifier parameters (ω). A malicious client also cannot guess the classifier parameters from the products returned as they are randomized by multiplication with r_{ji}. The reason why the server is able to compute the final sum S in *Step 8* of Algorithm 1 is because we impose the following condition on r_{ji}s and λ_js during its generation:

$$\forall_i, \sum_{j=1}^{k} \lambda_j \; r_{ji} = 1 \tag{1}$$

The privacy is based on the ability of the server to generate random numbers. We thus assume that the server has an access to a random number generator (PRNG). The λ_j and r_{ji} are generated using PRNG while ensuring that the Equation: 1 holds. This means that all but the last row of the r_{ji} and the corresponding λ_j are truly random. The last row of r_{ji} and λ_j are generated so as to satisfy the Equation: 1.

Algorithm 1 Authentication

```
1:  Client computes feature vector, $x_{1..n}$, from test data
2:  Each feature $x_i$ is encrypted ($E(x_i)$) and sent to server
3:  Server computes $kn+k$ random numbers, $r_{ji}$ and $\lambda_j$, such
    that, $\forall_i,\ \sum_{j=1}^{k} \lambda_j\ r_{ji} = 1$
4:  Server computes $E(\omega_i\ x_i\ r_{ji}) = E(\omega_i)\ E(x_i)\ E(r_{ji})$
5:  The $kn$ products thus generated are sent to the client
6:  The client decrypts the products to obtain: $\omega_i\ x_i\ r_{ji}$
7:  Client returns $S_j = \sum_{i=1}^{n} \omega_i\ x_i\ r_{ji}$ to the server
8:  Server computes $S = \sum_{j=1}^{k} \lambda_j\ S_j$
9:  if $S > \tau$ then
10:     return $Accepted$ to the client
11: else
12:     return $Rejected$ to the client
13: end if
```

Substituting the above equality in the expansion of the final sum (S) in Algorithm 1, we get:

$$S = \sum_{j=1}^{k} \lambda_j\ S_j = \sum_{j=1}^{k} \lambda_j \sum_{i=1}^{n} \omega_i\ x_i\ r_{ji} \tag{2}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{k} \lambda_j\ \omega_i\ x_i\ r_{ji} \tag{3}$$

$$= \sum_{i=1}^{n} \omega_i\ x_i \sum_{j=1}^{k} \lambda_j\ r_{ji} = \sum_{i=1}^{n} \omega_i\ x_i$$

We note that the server is unable to decipher any information about the original products in the whole process, and directly obtains the final sum-of-products expression. This quantity measures the confidence that the test biometric belongs to the claimed identity, and does not reveal any information about the actual biometric itself. The authentication process thus maintains a clear separation of information between the client and the server and hence provides complete privacy to the user, and security to the biometric. Moreover, the clear biometric or parameters are never stored at any place, thus avoiding serious losses if the server or the client computer is compromised. We will take a detailed look at the related security aspects in Section III. The extension of this approach to compute more complex functions such as the kernelized inner products are given in section IV. One can also deal with variable length features and warping based matching techniques using a similar approach. However, a complete treatment of such solutions are beyond the scope of this paper. We now look at the enrollment phase of the protocol.

B. Enrollment

In the previous section, we assumed that server has copies of the clients public key, E, as well as the classifier parameters that are encrypted using that key, $E(\omega_i)$. These were sent to the server during the enrollment phase by a trusted enrollment server. Assuming a third party as the enrollment server gives us a flexible model, where the enrollment could also be done by the client or the server if the trust allows.

During the enrollment, the client sends samples of her biometric to the enrollment server, who trains a classifier for the user. The trained parameters are encrypted and sent to the authentication server, and a notification is sent back to the client. Fig 3 gives an overview of the enrollment process. The biometric samples sent by the client to the enrollment server could be digitally signed by the client and encrypted using the servers public key to protect it.

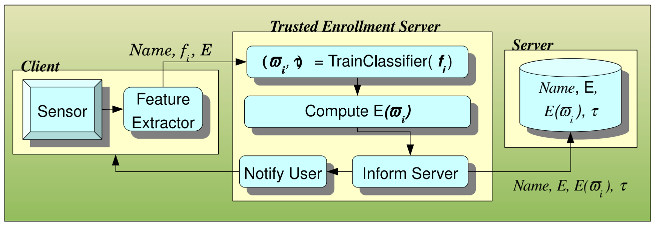

Trusted Enrollment Server
Client
Name, f_i, E
Sensor
Feature Extractor
(ϖ_i, τ) = TrainClassifier(f_i)
Compute E(ϖ_i)
Notify User
Inform Server
Server
Name, E, $E(\varpi_i)$, τ
Name, E, $E(\varpi_i)$, τ

Fig. 3. Enrollment based on a trusted third party(TTP): At the time of registering with a website, the encrypted version of the user's biometric template is made available to the website. The one-time classifier training is done on the plain biometrics, and hence requires a trusted server to handle training.

Algorithm 2 Enrollment

```
1: Client collects multiple sample of her biometric, $B_{1..k}$
2: Feature vectors, $x_i$, are computed from each sample
3: Client sends $x_i$, along with her identity and public key,
   $E$, to the enrollment server
4: Enrollment server uses $x_i$ and the information from other
   users to compute an authenticating classifier $(\omega, \tau)$ for the
   user
5: The classifier parameters are encrypted using the users
   public key: $E(\omega_i)$
6: $E(\omega_i)s$, along with the user's identity, the encryption key
   ($E$), and the threshold ($\tau$), are sent to the authentication
   server for registration
7: The client is then notified about success
```

The use of a third party for enrollment also allows for long-term learning by the enrollment server over a large number of enrollments, thus improving the quality of the trained classifier. Algorithm 2 gives a step-by-step description of the enrollment process. Note that the only information that is passed from the enrollment server to the authentication server is the users identity, her public key, the encrypted versions of the parameters, and a threshold value.

C. Applicability

We have not made any assumptions on the specific biometric being used in the framework. One could use any biometric as long as the feature vector embeds the samples in a Euclidean space. The classifier itself was assumed to be a linear classifier. However, one can extend it to work with kernel based methods (explained in section IV) and hence any verification problem that can be carried out using a generic SVM-based classifier can be modeled by this protocol. We also sketch an extension of the protocol that works with the Neural Networks in section IV.

III. Security, Privacy, and Trust in Blind Authentication

Security of the system refers to the ability of the system to withstand attacks from outside to gain illegal access or deny access to legitimate users. Since we are dealing with insecure networks, we are primarily concerned with the former. Security is hence a function of the specific biometric used as well as the overall design of the system. In terms of information revealed, security is related to the amount of information that is revealed to an attacker that would enable him to gain illegal access.

Privacy on the other hand is related to the amount of user information that is revealed to the server. Ideally, one would like to reveal only the identity and no additional information. Most of the current systems provide very little privacy, and hence demands trust between the user and the server. An ideal biometric system would ensure privacy and hence need not demand any trust, thus making it applicable in a large set of applications. We now take a closer look at the security and privacy aspects of the proposed system.

A. System Security

Biometric systems are known to be more secure as compared to passwords or tokens, as they are difficult to reproduce. As the authentication process in the proposed system is directly based on biometrics we gain all the advantages of a generic biometric system. The security is further enhanced by the fact that an attacker needs to get access to both the user's biometric as well as her private key to be able to pose as an enrolled user.

1) Server Security: We analyze the security at the server end using two possible attacks on the server:

Case 1: *Hacker gains access to the template database.* In this case, all the templates (or classifier parameters) in the server are encrypted using the public key of the respective clients. Hence gaining access to each template is as hard as cracking the public key encryption algorithm. Moreover, if by any chance a template is suspected to be broken, one could create another one from a new public-private key pair. As the encryption's are different, the templates would also be different. Brute-force cracking is practically impossible if one uses a probabilistic encryption scheme, even for limited-range data.

Case 2: *Hacker is in the database server* during *the authentication.* In such a situation, the hacker can try to extract information from his entire "view" of the protocol. Specifically, the view consists of the following five components:

1) Encrypted values of all ω_i's, that is $E(\omega_i)$, $i \in [1, n]$;
2) Encrypted values of all x_i's, that is $E(x_i)$, $i \in [1, n]$;
3) All the random values used in the protocol, that is all the r_{ji}'s, $i \in [1, n]$ and $j \in [1, k]$;
4) All the λ_j's, $j \in [1, k]$; and
5) All intermediate sums: $S_j = \left(\sum_{i=1}^{n} \omega_i x_i r_{ji}\right) \%N$ for all $j \in [1, k]$.

We ask, what can the hacker learn about the critical data, viz., ω_i's and x_i's? Note that the hacker only obtains k linear congruences over the n variables $y_1, y_2, \ldots, y_n$, namely, $S_j = \left(\sum_{i=1}^{n} r_{ji} y_i\right) \%N$ for all $j \in [1, k]$, where $y_i = \omega_i x_i$. Even though this may reveal some information about y_is, it is impossible to recover the original biometric, as it requires $|\mathbb{Y}|^{n-k}$ authentication trials ($|\mathbb{Y}|$ is domain of y_i's), each involving the help of the client and his private key. We now show that the amount of effort required in doing this is at least as much as randomly guessing the original biometric, and hence no additional information is revealed in principle.

Let $\mathbb{X}$ be the domain of x_i's and let $\mathbb{D}$ be the domain of r_{ji}'s. Without loss of generality, we assume that $\mathbb{D} \supset \mathbb{Y} \supset \mathbb{X}$, and all computations in the authentication protocol are done over the finite domain $\mathbb{D}$.

The number of authentication trials required in a brute-force attack of x_is is $O(|\mathbb{X}|^n)$, which is transformed to $O(|\mathbb{Y}|^{n-k})$ when the k linear congruences are revealed. We want to ensure that $|\mathbb{Y}|^{n-k} \geq |\mathbb{X}|^n$. That is, $ln(|\mathbb{Y}|) \geq \frac{n}{n-k} ln(|\mathbb{X}|)$. Solving this, we get:

$$k \leq n\left(1 - \frac{ln(|\mathbb{X}|)}{ln(|\mathbb{Y}|)}\right), \quad \text{or} \quad \frac{ln(|\mathbb{X}|)}{ln(|\mathbb{Y}|)} \leq 1 - \frac{k}{n}. \qquad (4)$$

We note that $|\mathbb{Y}|$ is around $|\mathbb{X}|^2$ as $y_i = x_i \omega_i$, which results in $k \leq n/2$ for complete privacy. As the minimum value of k that is required by the protocol is 2, we find that $2 \leq k \leq n/2$. Choosing a lower value of k will enhance security further, but increase the required $|\mathbb{D}|$.

Case 2.1: *If the hacker is in the server over multiple authentication trials of the same user*, then he will have multiple sets of k linear congruences to infer the values of y_i. However, note that the values of x_i will change slightly over multiple authentications, which gets reflected in the values of y_i. Now the hacker's problem is to compute an approximate estimate of y_i from his view of congruences over noisy y_is, which we call y'_i. Let $\varepsilon_i \in \mathbb{E}$ be the noise between the two instances of x_i. From linear algebra, we know that every additional set of k linear congruences will reduce the brute-force attack complexity by $O|\mathbb{Y}|^k$. Thus, it seems like after a certain number of authentication trials, a hacker will have sufficient congruences to uniquely solve for the n variables. However, we now show that even this is not possible, as during each authentication trial, the hacker not just obtains k additional equations but also ends up adding n new variables.

The hacker obtains k new equations in y'_i. As $y'_i = \omega_i(x_i + \varepsilon_i) = y_i + \omega_i \varepsilon_i$, this can be thought of as k new equations in y_i along with n new unknowns $\omega_i \varepsilon_i$. The domain of these new variables is $|\mathbb{E}|.|\mathbb{X}| \geq |\mathbb{X}|$. To ensure complete privacy, one has to make sure that the information gained by the additional k equations is less than the uncertainty introduced by the new n variables. That is, we need to ensure that $|\mathbb{Y}|^k \leq |\mathbb{X}|^n$. We also know, $|\mathbb{Y}|$ is around $|\mathbb{X}|^2$, thus we have to ensure that $|\mathbb{X}|^{2k} \leq |\mathbb{X}|^n$. This condition holds when $k \leq \frac{n}{2}$, which is true for any choice of k from the previous case. Thus, in spite of the view of multiple authentication trials, the hacker gets no additional information about the biometric.

Our scheme assumes that the server runs the delegated code faithfully. If the server is malicious, it can try to learn additional information about the client's biometric by using a selected vector (say unit vector in a direction) instead of the template for the product. However, the client can detect this using an input, whose result is known. For example, the client can randomly send a vector, which is known to be authentic (not authentic), and check if the the server accepts (rejects) it. Another option would be to use a probabilistic encryption scheme for the template, and keep the randomness in the encryption, a secret, as the server never needs to decrypt any data. In this case, the server will not be able to use any data other than the temple provided for computations.

Case 3: *Impostor trying blind attacks from a remote machine.*

It is clear that a brute force attack will have a complexity of the product of that of the plain biometric and the private key. However, note that in the final step, the computed confidence score S is a linear combination, and is compared with a threshold. Hence, if the impostor replaces the partial sums S_js with random numbers, he might be able to pass the confidence test without knowing anything about the biometric or the private key. Also note that the probability of success in this case could be very high. However, a simple modification of the protocol at the server side could thwart this attack. The server could multiply all the sums with a random scale factor, sf, and check if the returned sum is a multiple of sf or not. From his view, the impostor cannot learn sf as GCD is not defined for congruences.

In short, we see that the server is secure against any active or passive attack, and will not reveal any information about the classifier or the user's biometric.

2) Client Security: At the client side, we will consider the following attack scenarios:

Case 4: *Hacker gains access to the user's biometric or private key.*

Our protocol captures the advantages of both the biometric authentication as well as the security of the PKC. If the attacker gets hold of the user's biometric from external sources, he would also need the private key of the user to be able to use it. If only the private key of a user is revealed, the security for the effected individual falls back to that of using the plain biometric. Note that in practice, the private key is secured by storing it in a smart card, or in the computer using a fuzzy vault. In short, an impostor need to gain access to both the private key and the biometric to pose as a user. Even in this case, only a single user will be affected, and replacing the lost key would prevent any further damages. In practice, periodic replacement of the private key is advisable as in any PKC-based system.

Case 5: *Passive attack at the user's computer.*

In this case, the hacker is present in the user's computer during the login process. As the private key can be secured in a hardware which performs the encryption, the hacker will not have direct access to the private key. In other words, he will only learn the intermediate values of the computations. The hackers view will consist of kn quadratic congruences: $y_i r_{ji}, i \in [1, n], j \in [1, k]$ He further knows that there exists k λ_is that satisfy n congruences: $\sum_j \lambda_j r_{ji} \% N = 1$. Thus he has $kn + n$ quadratic congruences in $kn + n + k$ variables. This, as in case 2, results in an effort equivalent to a brute force attack. However if the hacker can stay in the user's computer over multiple authentications, then at some point of time, he will have sufficient number of congruences to solve for y_is (see case 2). Note that y_is does not reveal any useful information about the classifier. Moreover, any partial information gained is of no use as an authentication cannot be performed without access to the private key.

Note that an active attack in this case is identical to that of case 3, and the hacker does not know the private key.

3) Network Security: An insecure network is susceptible to snooping attacks. Let us consider the following attack scenarios:

Case 6: *Attacker gains access to the network.* An attacker who may have control over the insecure network can watch the traffic on the network, as well as modify it. The confidentiality of the data flow over the network can be ensured using the standard cryptographic methods like symmetric ciphers and digital signatures. Furthermore, all the traffic on the network are encrypted either using the clients public key or using the random numbers generated by the server. Hence, even if successfully snooped upon, the attacker will not be able to decipher any information. A replay attack is also not possible as the data communicated during the second round of communication is dependent on the random numbers generated by the server.

B. Privacy

Privacy, as noted before deals with the amount of user information that is revealed to the server, during the process of enrollment and authentication. We noted that there are two aspects of privacy to be dealt with:

1) *Concern of revealing personal information:* As the template or test biometric sample is never revealed to the server, the user need not worry that the use of biometrics might divulge any personal information other than her identity.
2) *Concern of being tracked:* One can use different keys for different applications (servers) and hence avoid being tracked across uses. In fact, even the choice biometric or real identity of the user itself is known only to the enrolling server. The authenticating server knows only the user ID communicated by the enrollment server and the biometric is obtained in the form of an encrypted feature vector.

As the user and server need not trust each other, the framework is applicable to a variety of remote and on-site identity verification tasks. Moreover, we note that there is no delegation of trust by the server to a program or hardware at the user's end, thus making it applicable to a variety of usage scenarios.

IV. Extension to Kernels and other Variations

Even though the linear classifier model can support some of the simple template matching approaches, it does not generalize to other model based classifiers. In the following subsections we will show the extensions for the proposed approach to deal with *a)* the kernel form of the linear classifier, the support vector machine (SVM), *b)* the neural networks, and *c)* the possible usability and the security extensions.

A. Kernel-based classification:

In the linear case, we described a procedure, $secureProduct$, to compute the inner product of two encrypted vectors without revealing its contents. However, in order to use a kernel based classifier at the server for verification, one needs to compute a discriminating function of the form:

$$S = \sum_{i=1}^{N} \alpha_i d_i \kappa({v_i}^T x) = \alpha \cdot \kappa(v, x), \tag{5}$$

where the rows of v are the support vectors and $\kappa()$ is referred to as the kernel function.

We first describe a simple extension of the $secureProduct$ procedure to deal with kernel based classification. We note that the parameter of the kernel function is a set of inner products of vectors. This could be calculated in a similar fashion as the regular blind authentication (using $secureProduct$). Once we obtain the individual inner products, we can compute the kernel functions, κ, at the server side. The discriminant function to be computed is once again the dot product of the vector of κ values and the α vector. This could again be computed, securely using the $secureProduct$ procedure. We note that this procedure allows us to compute any kernel function at the server side.

The above approach is more generic and secure than any of the secure authentication protocols in the literature. Moreover, it does not reveal any information about the classifier to the client. However, as the results of the intermediate inner products are known to the server, this simple extension is not completely blind in the information theoretic sense. This can be solved using another round of communication with the client and define a completely blind kernel-based verification protocol (as explained below).

Let the kernel function be $\kappa(v, x)$. Without loss of generality, we can model $\kappa()$ as an arithmetic circuit consisting of add and multiplication gates over a finite domain. Consider two encryption functions: E^* and E^+, which are multiplicative and additive homomorphic [4], [29], [30], respectively. The client has the private keys of both, while the public keys are available to the server also. We show that one can securely execute such a circuit using interaction between the server and the client. One can perform *addition* operations using $E^+()$ encrypted operands and *multiplication* operations using $E^*()$ encrypted operands, securely. The only cases of concern are when the operands of *multiplication* are in $E^+()$ and vice-versa. We show that if the server has $E^+(\mu)$ (encrypted using the public key of the client), it can convert it into $E^*(\mu)$ using one round of interaction with the client, without revealing μ to the client or the server. The details of the process are given in Algorithm 3.

Algorithm 3 $E^+(\mu)$ to $E^*(\mu)$

```
1: Initial State: The server has $E^+(\mu)$, and client has the corresponding private key.
2: The server chooses a random prime number $r$, and computes $E^+(\mu r)$ using repeated addition. This can be efficiently done in $O(log(r))$ additions using the well-known doubling technique.
3: The server sends $E^+(\mu r)$ to the client, who decrypts it to obtain $\mu r$, which reveals nothing about $\mu$.
4: The client then computes $E^*(\mu r)$ and sends this back to the server.
5: The server computes $E^*(\mu)$ by multiplying $E^*(\mu r)$ with $E^*(r^{-1})$.
```

Similarly, one may also want to convert $E^*(\mu)$ to $E^+(\mu)$. This is possible as explained in Algorithm 4. The above conversion procedures (described by Algorithms 3, 4) along with the secure product protocol (Algorithm 1) is sufficient for blind computation of any kernel based function such as radial basis function networks(RBFs). The computed confidence score S, is then compared by the server against the threshold τ to authenticate a user.

Algorithm 4 $E^*(\mu)$ to $E^+(\mu)$

```
1: Initial State: The server has $E^*(\mu)$, and client has the corresponding private key.
2: The server chooses a random prime number $r$, and computes $E^*(\mu r)$.
3: The server sends $E^*(\mu r)$ to the client, who decrypts it to obtain $\mu r$, which reveals nothing about $\mu$.
4: The client then computes $E^+(\mu r)$ and sends this back to the server.
5: The server computes $E^+(\mu)$ by repeatedly adding $E^+(\mu r)$, $r^{-1}$ times. This can be efficiently done in O(log($r^{-1}$)) additions using the well known doubling technique.
```

For example, consider a polynomial kernel, $\kappa(v, x) = ({v_i}^T \cdot x)^p$, that is to be securely computed in our setting. Initially, the server has access to the encrypted feature vector $\vec{x}$ and the encrypted support vectors $\vec{s}v_k$. The initial encryption scheme is assumed to be multiplicative homomorphic. Now, computing the kernel value requires both addition and multiplication operations among the support vectors and the feature vector. Utilizing the switch encryption protocols 3 and 4, the polynomial kernel can be computed by using two rounds of switch operations per support vector. The final confidence score S is computed using the secure dot product protocol 1.

The complete protocol to securely compute a polynomial kernel is shown in Figure 4.

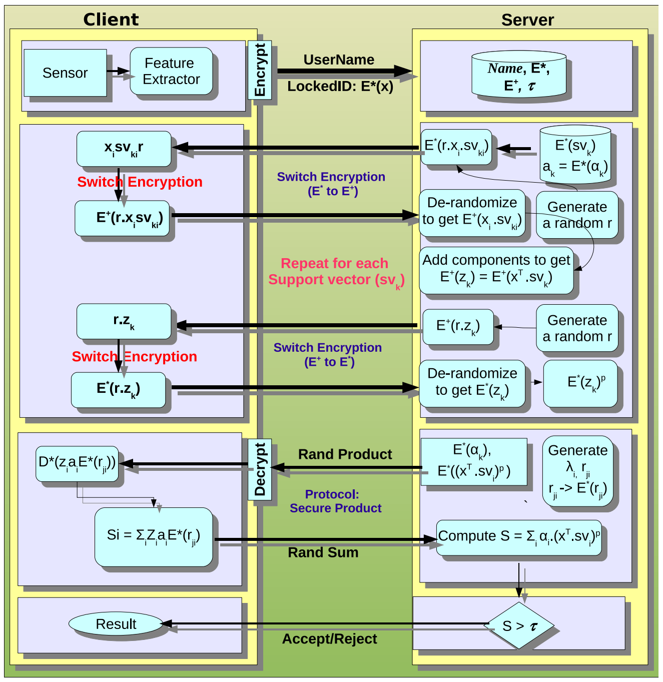

Client
Server
Sensor
Feature Extractor
Encrypt
UserName
LockedID: E*(x)
Name, E*, E+, τ
$x_i sv_{ki} r$
$E^*(r.x_i.sv_{ki})$
$E^*(sv_k)$ $a_k = E^*(\alpha_k)$
Switch Encryption
Switch Encryption (E^* to E^+)
$E^+(r.x_i sv_{ki})$
De-randomize to get $E^+(x_i.sv_{ki})$
Generate a random r
Repeat for each Support vector (sv_k)
Add components to get $E^+(z_k) = E^+(x^T.sv_k)$
$r.z_k$
$E^+(r.z_k)$
Generate a random r
Switch Encryption
Switch Encryption (E^+ to E^*)
$E^*(r.z_k)$
De-randomize to get $E^*(z_k)$
$E^*(z_k)^p$
$D^*(z_i a_i E^*(r_{ji}))$
Decrypt
Rand Product
$E^*(\alpha_k)$, $E^*((x^T.sv_i)^p)$
Generate λ_i, r_{ji} r_{ji} -> $E^*(r_{ji})$
Protocol: Secure Product
$Si = \Sigma_i Z_i a_i E^*(r_{ji})$
Rand Sum
Compute $S = \Sigma_i \alpha_i.(x^T.sv_i)^p$
Result
Accept/Reject
S > τ

Fig. 4. Blind authentication process for a polynomial kernel.

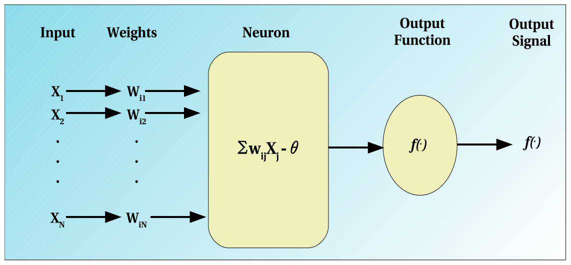
In general, the computed confidence score may be considered as an input to a new classifier. For example, in neural networks, the output at one layer is passed as input to the next layer. In such scenarios, one may wish to keep the server oblivious of the computed score S. Thus, we define a *Blind Secure Product Protocol*, Algorithm 5, that computes only the encryption of the score S.

Algorithm 5 Blind Secure Product Protocol

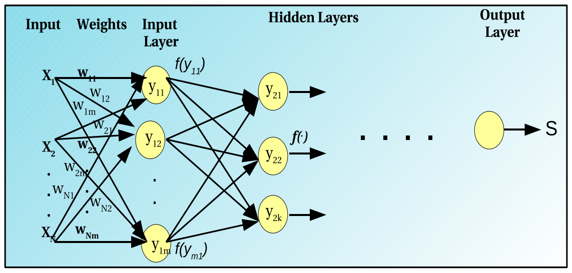
```
1: Initial State: The server has $E^*(\omega)$, $E^*(x)$ received from the client.
2: Server computes $kn+k$ random numbers, $r_{ji}$ and $\lambda_j$, such that, $\forall_i$, $\sum_{j=1}^{k} \lambda_j \ r_{ji} = 1$
3: Server computes $E(\omega_i \ x_i \ r_{ji}) = E(\omega_i) \ E(x_i) \ E(r_{ji})$
4: The $kn$ products thus generated are sent to the client
5: The client decrypts the products to obtain: $\omega_i \ x_i \ r_{ji}$
6: Client computes $S_j = \sum_{i=1}^{n} \omega_i \ x_i \ r_{ji}$
7: $S_j$ is encrypted using $E^+$ and $E^+(S_j)$ is send over to the server.
8: Server computes $E^+(S) = \sum_{j=1}^{k} \sum_{i=1}^{\lambda_j} E^+(S_j)$, this can be efficiently computed using the well known doubling technique.
```

B. Neural Network based classification

The generalization and approximation provided by Neural Networks have presented them as a practical method for learning real-valued, discrete-valued and vector-valued functions. ANN learning is well-suited to problems in which the training data corresponds to noisy, complex sensor data, such as inputs from cameras [31], thus making them ideal candidate for applications in biometric classification/verification.



(a)



(b)

Fig. 5. *a)* A typical processing unit used as a node in ANN. A weighted summation of the input is computed, result of which is then used to computed the output function $f()$, *b) A Typical Multilayer Neural Network.*

Over the years a large number of methods based on Neural Networks has been proposed for biometric verification [23], [32]–[34]. In this section, we show how our proposed protocol is generic enough to blindly and securely evaluate a neural network.

A neural network [25] consists of simple processing elements called neurons [Fig: 5 (a)], which consists of a summing part and an output part. The summing part computes a weighted summation of the input vector whereas the output function determines the output signal. An ANN is made up of various layers [Fig: 5 (b)] the first layer being called *the input layer*, and the last layer *the output layer*, the rest being known as *hidden layers*. Each layer, have a pre-defined number of neurons, computes a weighted summation of the input to it and generates an output signal, which becomes an input to the next layer.

Threshold and Sigmoid are the two most popular type of basic units used in ANN. A perceptron is same as the linear classifier discussed in section II. It takes a vector of real-valued inputs, calculates a weighted summation of these inputs and outputs a 1 if result is greater than the threshold and -1 otherwise. Algorithm 6 describes the completely blind perceptron computation.

$$S = sgn(y) = \begin{cases} 1 & \text{if } y \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (6)$$

Another important/popular basic unit in ANN is the *Sigmoid Unit*. It is based on a smoothed, differential threshold function.

Algorithm 6 Blind Threshold Function Computation

```
1: Initial State: The server has $E^*(\mu)$, $E^*(x)$ received from the client. Server to compute $E^*(t)$, where t = 0/1 depending on threshold.
2: After a round of *Blind Secure Product Protocol* [Algo: 5], the server obtains $E^+(\mu^T.x - \alpha)$
3: Server generates a random number $r$ and computes $E^*(r(\mu^T.x - \alpha)$ and sends over to the client.
4: Client decrypts the obtained cipher and returns back the encrypted equivalent of sign bit i.e. returns $E^*(d)$ = $E^*(sign(r(\mu^T.x - \alpha)$
5: Server computes $E^*(S)$ = $E^*(d).E^*(sign(r))$
```

The sigmoid unit first computes a linear combination of its inputs, then applies a threshold to the result. The threshold output is a continuous function of its input, Equation 7.

$$S = \sigma(y) = \frac{1}{1 + e^{-\alpha.y}} \tag{7}$$

The α, in the above equation, is some positive constant that determines the steepness of the threshold. A completely blind Sigmoid function computation is explained in Algorithm 7.

Algorithm 7 Blind Sigmoid Function Computation

```
1: Initial State: The server has $E^*(\mu)$, $E^*(x)$ received from the client. Server to compute $E^*(\frac{1}{1+e^{-\alpha.y}})$.
2: After a round of *Blind Secure Product Protocol* [Algo: 5], the server obtains $E^+(y)$
3: $E^+(\alpha.y)$ is computed using repeated additions.
4: Server chooses a random $r$ and sends to client $E^+(r+\alpha.y)$ = $E^+(r).E^+(\alpha.y)$.
5: Client decrypts the obtained cipher to get $r + \alpha.y$, which is used to compute $E^*(e^{r+\alpha.y})$ and is sent back to the server.
6: Server multiplies the obtained result with $E^*(e^{-r})$ to get $E^*(e^{\alpha.y})$.
7: Switch encryption and add $E^+(1)$ to obtain $E^+(1 + e^{\alpha.y})$
8: Server chooses a random $r = \frac{r_1}{r_2}$, such that $r^{-1}$ exists. Use repeated additions to obtain, $E^+(r_1.e^{\alpha.y})$ and $E^+(r_2.e^{\alpha.y+1})$. These are then send over to the client.
9: Client decrypts the received ciphers and computes $r.\frac{e^{\alpha.y}}{1+e^{\alpha.y}}$. This is encrypted using $E^*$ and send over to server.
10: Server obtains $E^*(\frac{1}{1+e^{-\alpha.y}})$ by multiplying $E^*(r.\frac{1}{1+e^{-\alpha.y}})$ and $E^*(r^{-1})$.
```

With the solutions already sketched for securely computing both sigmoid and perceptron based neurons, the solution can be easily extended to securely compute multilayer neural networks. A typical multilayer neural network is shown in Fig 5 (b).

Every neuron in each of the layers is securely computed using algorithms already described. In the process, the client doesn't learn anything and all that the server gets is the encrypted output of the neuron. This encrypted output of a particular layer of neurons acts as an input to the next layer in the network. The output of the last layer is decrypted and compared against the threshold to authenticate the user.

The above process is completely secure and blind in that at no point does the server or client learns the weights or intermediate results. All computations are done in encrypted domain, and given an encrypted input vector $E^*(x)$ the client learns nothing but the authentication result. A somewhat similar solution was proposed by Orlandi *et al* [35], however, their solution uses only additive homomorphic encryption schemes and is therefore not as generic as the one proposed by us. Moreover, their solution assumes the hidden layer weights are available in plain with the server, thus compromising both the security and privacy of the system.

C. Usability and Security Extensions

One could extend the basic blind authentication protocol in a variety of ways to improve the usability and security.

Client side security: The users client module (computer) contains the public and private keys for encryption and decryption. Moreover the client end also contains the biometric acquisition device. To ensure complete security of the system, one needs to consider the security at the client end also. This is especially true, if one is using a public terminal to access any service. The first step in securing the private key is to move it to a card so that the private key is not lost if the client computer is compromised. As a second step one could carry out the decryption operation, completely in a smart card. Revealing the secret keys to an attacker can reduce the overall security of the system to that of a plain biometric authentication system.

One could also secure the secret keys at the client end using a fuzzy vault [20], either in the client's computer or on a card. The biometric that is provided for authentication can also be used to unlock the vault to get the key. The released private key is used for decryption of results in the protocol. The fuzzy vault construct precisely suits this purpose as one could blindly use the keys generated by unlocking the vault for encryption. If the biometric presented is wrong, the encryption will not match the server's keys and hence the authentication will fail. Hence we have a double layer of security through the biometric provided by the user.

Avoiding client-side computation and communication: Another possible extension to the framework is to use the paradigms from secure computing to package the intermediate operations done at the client side into an applet. This applet can now be run securely on the server itself, thus avoiding the overhead of communication, and reducing the computing requirements of the client.

Using different encryption schemes: Note that RSA is just one of the public key encryption algorithms that is homomorphic to multiplication. We could replace this with any of the other similar encryption mechanisms. One could analyze the computation cost and security issues for each encryption method.

Since the information content in each feature (or weight) is expected to be limited and the public key of the client is known, it may be possible for an attacker to decode the encrypted features (weights) using a direct plain-text attack.

Similarly in the blind threshold function computation, output of the neuron is either zero or one. To combat this attack, public key encryption schemes must incorporate an element of randomness, ensuring that each plaintext maps into one of a large number of possible ciphertexts. Thus, the encryption scheme $E()$ has to be a function of both the secret x and a random parameter r. Such a scheme is known as *probabilistic encryption*. However, for our purpose, we also need to carry out the computations in the encrypted space, thus the encryption scheme should also be homomorphic. ElGamal [29] and Pailler Encryption [30] are two popular probabilistic homomorphic encryption schemes.

Improving speed of SVM-based classifiers: As described in Section IV, the kernel based classifiers need to compute the discriminating function given by Equation 5. As can be noticed, the computational costs of computing this is directly proportional to the number of support vectors used. In practice, the number of support vectors that are returned from the training step could be quite large. However, a variety of approaches to reduce the number of support vectors used (without loss in accuracy) for classification has been proposed [24].

V. Implementation and Analysis

We have performed several experiments to evaluate the efficiency and accuracy of the proposed approach. An authentication protocol was implemented based on a client-server model that can perform verification over an insecure channel such as the Internet. A variety of public domain datasets are evaluated using an SVM classifier to demonstrate the effectiveness of our proposed protocol. The following experiments and analysis evaluates the accuracy and performance of the proposed approach for verification.

A. Implementation

For the evaluation purpose a generic SVM based verifier based on a client-server architecture was implemented in GNU/C. RSA keys were generated using the implementation available through *XySSL* [36] and keys for the Paillier cryptosystem were generated using the *Paillier Library* [37] . All mathematical computations were done using the *GNU Multiple Precision Arithmetic Library (GMP)* [38]. All experiments are conducted on AMD X2 Dual Core 4000+ processor, 750MB DDR2 RAM and 100Mbps Intranet.

Both RSA and Paillier cryptosystem have exponentiation based encryption and decryption. Their implementation assumes that the data consists of positive integers. For the homomorphism to hold, we need to map the floating point numbers to positive integers. Hence we scale the feature vectors and the SVM parameters to retain the precision and round off to the nearest integral value. Efficiently handling negative numbers is important to achieve efficiency. The representation chosen should ensure a single representation of zero, obviating the subtleties associated with negative zero. In our implementation, the mathematical library operates at the binary representation level. We use an implicit sign representation to handle negative numbers. If the range of numbers used is $(0, M)$, then we use the numbers in the range $(0, M/2)$ to represent positive numbers, and for the remaining numbers negative. For example: let $M = 256$, then to represent -95 we store $-95 modulo 256$ which is equivalent to 161 since: $-95 + 256 = -95 + 255 + 1 = 160 + 1 = 161$

If x_i is a parameter to be encrypted, the forward mapping is defined as: $x'_i = fwdMap(\lfloor s.x_i + 0.5 \rfloor)$, where s is a scale factor, depending on the range of values for x_is, and $fwdMap()$ maps the integral numbers to the implicit sign representation. The corresponding reverse mapping is done by the server, once the results are obtained.

In the following sub-sections, we will validate the generality of the protocol by validating classification of various publicly available datasets. We will also analyze how the various parameters i.e. key-size, precision affect the classification accuracy and the verification time. Finally we'll show the validity of SVM's as a classification model for various biometric problems.

B. Classification Accuracy

As the protocol implements a generic classifier, without making any simplification assumptions, the accuracy of the classifier should be identical to that of the original classifier. One could expect small variations in accuracy due to the round off errors used in the mapping function described above. To verify the effect we compared the classification results using linear and SVM classifiers of 8 different public domain datasets: the *Iris*, *Liver Disorder*, *Sonar*, *Diabetes*, and *Breast Cancer* datasets from the UCI repository and the *Heart* and *Australian* datasets from the Statlog repository. The datasets were selected to cover a variety of feature types and feature vector lengths. Table II describes the datasets and the accuracy obtained using a polynomial kernel with precision set as 4. On these datasets, the classification results remained identical even though there were minor variations in the computed discriminant values.

Dataset	Number of Features	Number of Instances	Accuracy (%)
Iris [UCI]	4	150	100
Heart [Statlog]	13	270	90
Liver Disorder [UCI]	6	345	68
Sonar [UCI]	60	208	51.47
Australian [Statlog]	14	690	86.49
Diabetes [UCI]	8	768	76.37
FourClass [Tin Kam Ho]	2	862	69.20
Breast Cancer [UCI]	10	683	89.80

TABLE II
Classification results on various datasets using a SVM classifier. The accuracies were compared to the corresponding plain domain classifier and was found to be identical.

The above accuracies were cross checked by re-classifying the datasets with the same parameters by the well known SVM classification library SVM^{light} [39]. Figure 6 shows the verification time for a linear classifier w.r.t. various RSA key-sizes and feature vector lengths. A more detailed analysis of the computational time for the protocol is given in Section V-D.

Figure 7 shows how the overall accuracy is affected by changing the precision. For the considered datasets, the feature

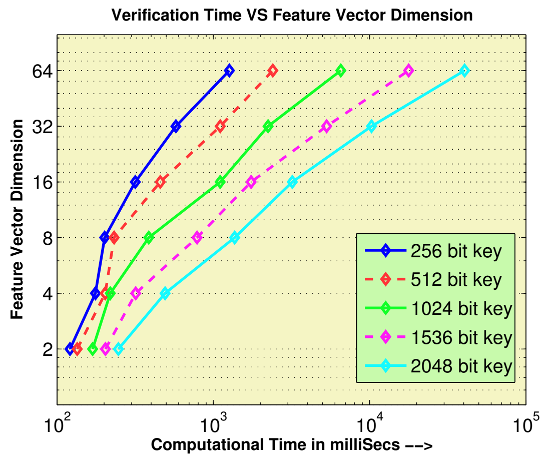


Fig. 6. Verification time for various key sizes and feature vector lengths.

vectors were first normalized to range -1 to 1 and then scaled to retain a certain precision. When precision is set to less than 2, a lot of feature vectors having feature values of the order of 10^{-3} or less, mapped to a value of zero, thus affecting the accuracy. For the above datasets, we note that a precision of 3 or more results in stable results and the accuracies do not change with any further increase in precision. Thus for our experiments we set precision as 4. Note: precision doesn't affect the computational time, as all the numbers are represented using a fixed length bit representation.

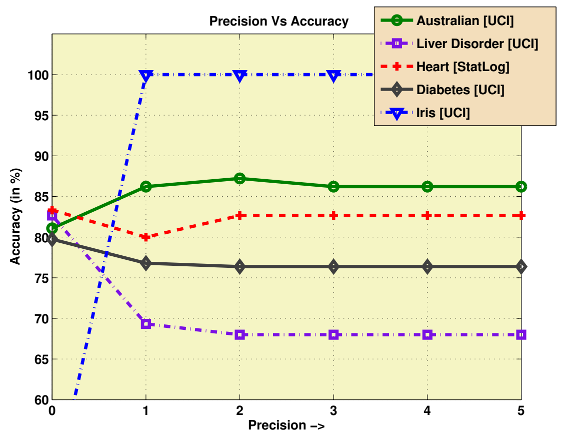


Fig. 7. Variation of accuracy w.r.t. the precision of representation.

The above set of experiments demonstrate the applicability of our protocol to the SVM based classification problems. We showed that one can achieve the accuracies of SVM's even in an encrypted domain and at the same time obtain heightened security at some computational expense.

C. Biometric Verification

We have presented a protocol that is able to securely classify data using Support Vector Machines and Neural Networks (Section IV). The primary limitation of the protocol in its current form is its restriction to fixed length feature vector representation of the data (Section II). However, we note that there are techniques that employ fixed length representation of various biometrics with performances that are comparable those using variable length representations. Some of well known matching techniques using variable length features are graph based local structure representation of minutiae by Kisel *et al* [40], Time series representation of hand geometry by Vit *et al* [41], Face Bunch Graph representation of face by Wiskott *et al* [42]. Comparable accuracies have been reported using fixed length representation such as the invariant moment features for fingerprints by Yang *et al.* [43], the hand geometry features by David *et al.* [44], the 3-D morphable face model by Blanz *et al.* [45], and the DCT coefficient representation of the Iris by Monro *et al.* [46], all achieve performances close to the state of the art in the respective biometrics.

To verify the effectiveness of using SVMs as a classification model for biometric verification problems, we tested it on four different modalities. The verification accuracies after 3-fold cross validation on each of the datasets is presented in Table III.

- The first set of experiments used Eigen face representation as features on the Yale face dataset [47], consisting of 10 users, with 11 samples for each user. For each experiment 4 samples were used for training and the remaining 7 samples were used for testing.
- For the second set of experiments, we used a hand-geometry data-set that was collected in-house. The data-set consisted of 149 users with 10 hand images each. The features consists of the 14 finger length and width features described by Jain *et al.* [48]. For each experiment 4 images per user were used for training purpose and the remaining 6 were used for testing.
- The third were on the CASIA IRIS database [49]. The Version 1 of the data-set consists of 108 users with 7 images per user (the seven images are collected over two separate imaging sessions). The iris code consists of 9600 binary features. 3 samples per user were used for training and 4 sample per user were used for testing purpose in each experiment.
- The forth and the final data-set used was *Fingerprint Verification Contest 2004 (FVC2004* data-set [50]. The DB2_A data-set consists of 100 users with 8 images per user. 7 invariant moment features are used as the feature vector. 3 images per user are used for training purpose and the remaining 5 used for testing for each experiment.

Dataset	# of Features	Avg num of Support Vectors	Accuracy
Hand Geometry	20	310	98.38%
Yale Face	102	88	96.91%
CASIA Iris	9600	127	98.24%
FVC 2004	7	440	84.45%

TABLE III
VERIFICATION ACCURACY ON BIOMETRIC DATASETS.

Figure 8 shows the *receiver operating characteristic (ROC)* [51] plots for the biometrics using fixed length representation[2]. The primary objective of the experiments is to demonstrate that making the authentication secure *does not decrease* the accuracy. Hence, one can apply the technique

[2]* Yang *et al* [43], **Wang *et al.* [52]

to secure any fixed-length representation of a biometric trait, which is classified using an SVM or Neural Network.

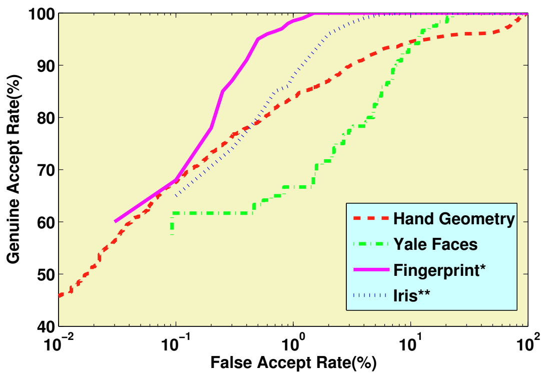


Fig. 8. ROC Curves for verification

D. Computation and Communication Overheads

The additional computation that needs to be carried out can be divided into two parts: i) Modulo multiplications to be done for encryption/decryption and inner product, and ii) the additional time spent in the computation of random numbers, products and sums. As the modulo multiplications and encryption decryption operations can be done efficiently using dedicated hardware available [53], we analyze the time required for both, separately. Consider a biometric with feature vector of length n. In the protocol, the client needs to do n encryptions for the test vector x.

For the linear classifier, the server needs to do kn encryptions of the random numbers and $2kn$ multiplications, so as to compute $E(\omega_i x_i r_{ji})$, where k≤n . The client needs to do kn decryptions. Additional computations at the server includes $n+kn$ modulo multiplications of encrypted numbers at the server end, and kn non-encrypted additions at the client end. In addition, the server generates kn random numbers. For most practical biometrics, the total run time required for all these (non-encrypted) computations together on current desktop machines is less than 10 milliseconds. The communication overhead, in addition to regular authentication, includes sending kn numbers from the server to the client and sending k numbers from the client back to the server for evaluation of the final result.

Extending the analysis to a direct kernel based classifier with n_v support vectors (SV), one need to repeat the *secure product* n_v times, once for every SV. Another round of *secure product* computes the final result. Hence the time required will be $n_v + 1$ times that required for the linear classifier. In practice the total time taken (other than those implemented in hardware) is less than one second.

For the completely blind kernel-based protocol, the first phase is the same as the direct kernel extension. However, to achieve complete blindness, we need to do one round of communication to switch encryptions, that will include a kn_v length vector to be sent from the server to the client and back. In the third phase, the computation and communication is identical to that required for a single *secure product*. Hence the total time required will be $n_v + 2$ times that required for the linear classifier.

One could achieve further computational efficiency through support-vector reductions, as well as employing other more computationally fast homomorphic encryption schemes.

VI. Discussions and Conclusions

The primary advantage of the proposed approach is that we are able to achieve classification of a strongly encrypted feature vector using generic classifiers such as Neural Networks and SVMs. In fact, the authentication server need not know the specific biometric trait that is used by a particular user, which can even vary across users. Once a trusted enrollment server encrypts the classifier parameters for a specific biometric of a person, the authentication server is verifying the identity of a user with respect to that encryption. The real identity of the person is hence not revealed to the server, making the protocol, completely blind. This allows one to revoke enrolled templates by changing the encryption key, as well as use multiple keys across different servers to avoid being tracked, thus leading to better privacy.

The proposed blind authentication is extremely secure under a variety of attacks and can be used with a wide variety of biometric traits. Protocols are designed to keep the interaction between the user and the server to a minimum with no resort to computationally expensive protocols such as SMC [54]. As the verification can be done in real-time with the help of available hardware, the approach is practical in many applications. The use of smart cards to hold encryption keys enables applications such as biometric ATMs and access of services from public terminals. Possible extensions to this work includes secure enrollment protocols and encryption methods to reduce computations. Efficient methods to do dynamic warping based matching of variable length feature vectors can further enhance the utility of the approach.

References

[1] A. K. Jain, A. Ross, and S. Prabhakar, “An introduction to biometric recognition,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, January 2004.
[2] N. K. Ratha, J. H. Connell, and R. M. Bolle, “Enhancing security and privacy in biometrics-based authentication systems,” *IBM Systems Journal*, vol. 40, no. 3, pp. 614–634, Mar. 2001.
[3] “Proceedings of Worshop on Biometrics (CVPR),” 2006,07.
[4] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[5] C. Fontaine and F. Galand, “A survey of homomorphic encryption for nonspecialists,” *EURASIP*, vol. 1, pp. 1–15, 2007.
[6] C. Gentry, “Fully homomorphic encryption using ideal lattices,” *STOC*, pp. 169–178, 2009.
[7] F. Farooq, R. M. Bolle, T.-Y. Jea, and N. Ratha, “Anonymous and revocable fingerprint recognition,” in *CVPR Biometrics Worshop*, Jun. 2007, pp. 1–7.
[8] A. Teoh, D. Ngo, and A. Goh, “Biohashing: Two factor authentication featuring fingerprint data and tokenised random number,” *Pattern Recognition*, vol. 37, no. 11, pp. 2245–2255, November 2004.
[9] A. K. Jain, K. Nandakumar, and A. Nagar, “Biometric template security,” *EURASIP*, vol. 8, no. 2, pp. 1–17, 2008.
[10] M. Upmanyu, A. M. Namboodiri, K. Srinathan, and C. V. Jawahar, “Efficient biometric verification in the encrypted domain,” in *Third International Conference on Biometrics*, Jun. 2009, pp. 906–915.
[11] U. Uludag, S. Pankanti, S. Prabhakar, and A. K. Jain, “Biometric cryptosystems: Issues and challenges,” *Proceedings of the IEEE*, vol. 92, no. 6, pp. 948–960, Jun. 2004.

[12] N. Ratha, S. Chikkerur, J. Connell, and R. Bolle, “Generating cancelable fingerprint templates,” *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, vol. 29, no. 4, pp. 561–572, Apr. 2007.

[13] A. Teoh, B. Jin, T. Connie, D. Ngo, and C. Ling, “Remarks on BioHash and its mathematical foundation,” *Information Processing Letters*, vol. 100, no. 4, pp. 145–150, Nov. 2006.

[14] M. Savvides and B. Vijaya Kumar, “Cancellable biometric filters for face recognition,” *International Conference on Pattern Recognition (ICPR)*, vol. 3, pp. 922–925, 2004.

[15] A. Kong, K. Cheung, D. Zhang, M. Kamel, and J. You, “An analysis of biohashing and its variants,” *Pattern Recognition*, vol. 39, no. 7, pp. 1359–1368, July 2006.

[16] T. Connie, A. Teoh, M. Goh, and D. Ngo, “PalmHashing: a novel approach for cancelable biometrics,” *Information Processing Letters*, vol. 93, no. 1, pp. 1–5, Jan. 2005.

[17] T. Boult, W. Scheirer, and R. Woodworth, “Revocable fingerprint biotokens: Accuracy and security analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2007, pp. 1–8.

[18] A. K. Jain and U. Uludag, “Hiding biometric data,” *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, vol. 25, no. 11, pp. 1494–1498, Nov. 2003.

[19] W. J. Scheirer and T. E. Boult, “Bipartite biotokens: Definition, implementation, and analysis,” in *International Conference on Biometrics (ICB)*, 2009, pp. 775–785.

[20] A. Juels and M. Sudan, “A fuzzy vault scheme,” *Designs, Codes and Cryptography*, vol. 38, no. 2, pp. 237–257, 2006.

[21] Y. Dodis, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” in *Eurocrypt*, 2004, pp. 523–540.

[22] T. E. B. Walter J. Scheirer, “Cracking fuzzy vaults and biometric encryption,” in *Biometrics Symposium*, 2007.

[23] K. Nagai, H. Kikuchi, W. Ogata, and M. Nishigaki, “ZeroBio: Evaluation and development of asymmetric fingerprint authentication system using oblivious neural network evaluation protocol,” in *The Second International Conference on Availability, Reliability and Security (ARES)*, Apr. 2007, pp. 1155–1159.

[24] S. Abe, *Support Vector Machines For Pattern Classification*. Springer, 2005.

[25] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[26] S. Avidan and M. Butman, “Blind vision,” in *European Conference on Computer Vision (ECCV)*, 2006, pp. 1–13.

[27] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[28] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, Oct. 1996.

[29] T. El Gamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[30] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” *Eurocrypt*, pp. 223–238, 1999.

[31] T. Mitchell, *Machine Learning*. The McGraw-Hill, 1997.

[32] A. Ceguerra and I. Koprinska, “Automatic fingerprint verification using neural networks,” in *International Conference on Artificial Neural Networks (ICANN)*, 2002, pp. 1281–1286.

[33] M. Faundez-Zanuy, D. A. Elizondo, M. Ángel Ferrer-Ballester, and C. M. Travieso-González, “Authentication of individuals using hand geometry biometrics: A neural network approach,” *Neural Process. Lett.*, vol. 26, no. 3, pp. 201–216, 2007.

[34] H. El-Bakry, “Fast iris detection for personal verification using modular neural nets,” in *Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications*, 2001, pp. 269–283.

[35] C. Orlandi, A. Piva, and M. Barni, “Oblivious neural network computing via homomorphic encryption,” *EURASIP*, vol. 2007-1, pp. 1–10.

[36] “Xyssl,” http://linux.softpedia.com/get/Security/XySSL-19360.shtml.

[37] J. Bethencourt, “Paillier library,” http://acsc.csl.sri.com/libpaillier/.

[38] “Gnu multiple precision arithmetic library,” http://gmplib.org/.

[39] T. Joachims, “Svm-light,” http://svmlight.joachims.org/.

[40] A. KiselL, A. Kocochetkov, and J. Kranauskas, “Fingerprint minutiae matching without global alignment using local structures,” *INFORMATICA*, vol. 19, no. 1, pp. 31–44, 2008.

[41] V. N. W. A. Ratanamahatana, “Hand geometry verification using time series representation,” Sep 2007.

[42] L. Wiskott, J.-M. Fellous, N. Krueuger, and C. von der Malsburg, *Face Recognition by Elastic Bunch Graph Matching*, ser. Intelligent Biometric Techniques in Fingerprint and Face Recognition. CRC Press, 1999.

[43] J. Yang, jinWook Shin, B. Min, J. Park, and D. Park, “Fingerprint matching using invariant moment fingercode and learning vector quantization neural network,” *ICCIS*, vol. 1, pp. 735–738, Nov 2006.

[44] M. Faundez-Zanuy, D. A. Elizondo, M. Ángel Ferrer-Ballester, and C. M. Travieso-González, “Authentication of individuals using hand geometry biometrics: A neural network approach,” *Neural Process. Lett.*, vol. 26, no. 3, pp. 201–216, 2007.

[45] V. Blanz and T. Vetter, “Face recognition based on fitting a 3d morphable model,” *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, vol. 25-9, pp. 1063–1074, 2003.

[46] D. M. Monro, S. Rakshit, and D. Zhang, “Dct-based iris recognition,” *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, vol. 29, no. 4, pp. 586–595, 2007.

[47] “Yale face database,” http://cvc.yale.edu/projects/yalefaces/yalefaces.html.

[48] A. K. Jain, A. Ross, and S. Pankanti, “A prototype hand geometry-based verification system,” in *In 2nd Int’l Conference on Audio- and Video-based Biometric Person Authentication (AVBPA)*, 1999, pp. 166–171.

[49] “Casia iris dataset,” http://www.cbsr.ia.ac.cn/english/Databases.asp.

[50] “Fvc2004 dataset,” http://bias.csr.unibo.it/fvc2004/databases.asp.

[51] T. Fawcett, “An introduction to roc analysis,” *Pattern Recogn. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[52] Y. Wang and J. Han, “Iris recognition using support vector machines,” in *ISNN (1)*, 2004, pp. 622–628.

[53] T. Blum and C. Paar, “High-radix montgomery modular exponentiation on reconfigurable hardware,” *IEEE Transactions on Computers*, vol. 50, no. 7, pp. 759–764, 2001.

[54] A. C.-C. Yao, “How to generate and exchange secrets,” *Foundations of Computer Science*, pp. 162–167, 1986.

Maneesh Upmanyu received his BTech (CSE) from IIIT, Hyderabad and is pursuing his Masters degree at the Center for Visual Information Technology (CVIT) at IIIT-H.

His research interests include Computer Vision, Machine Learning, Cryptography and Information Security.

Anoop Namboodiri received his PhD from Michigan State University in 2004. Prior to this, he worked with Siemens Communication Systems, and the Center for AI and Robotics (CAIR) till 1999. He is currently an Assistant Professor at IIIT, Hyderabad, working with the Center for Visual Information Technology.

His research interests include Biometrics, Pattern Recognition, and Computer Vision.

Kannan Srinathan received his MS in CSE and PhD from IIT, Madras in 2001, and 2007 respectively. He is currently an Assistant Professor at IIIT, Hyderabad, working with the Center for Security, Theory and Algorithmic Research (CSTAR).

His research interests include Security, Distributed computing, Cryptography, and Algorithms.

C.V. Jawahar received his PhD from IIT Kharagpur, India. He worked with Center for Artificial Intelligence and Robotics, Bangalore till Dec 2000, and since then he is with IIIT Hyderabad, India. He is presently a Professor at IIIT.

His does research in the broad area of Computer Vision and Multimedia Systems.