Efficient Biometric Verification in Encrypted Domain

Maneesh Upmanyu, Anoop M. Namboodiri, K. Srinathan, and C.V. Jawahar

Center for Visual Information Technology
International Institute of Information Technology, Hyderabad
{upmanyu@research.,anoop@,srinathan@,jawahar@}iiit.ac.in

Abstract. Biometric authentication over public networks leads to a variety of privacy issues that needs to be addressed before it can become popular. The primary concerns are that the biometrics might reveal more information than the identity itself, as well as provide the ability to track users over an extended period of time. In this paper, we propose an authentication protocol that alleviates these concerns. The protocol takes care of user privacy, template protection and trust issues in biometric authentication systems. The protocol uses asymmetric encryption, and captures the advantages of biometric authentication. The protocol provides non-repudiable identity verification, while not revealing any additional information about the user to the server or vice versa. We show that the protocol is secure under various attacks. Experimental results indicate that the overall method is efficient to be used in practical scenarios.

1 Introduction

The primary advantages of biometrics over other authentication mechanisms are its convenience, security, and non-repudiable nature [1]. However, the assertions on security and non-repudiation are valid only if the integrity of the overall system is maintained [2] A hacker who gains physical or remote access to an authentication server can steal the stored templates, which are non-replaceable in case of plain templates. Concerns are also on the privacy as many biometrics reveal personal information beyond just identity. Widespread use of biometric authentication also provides the ability to track a person through every activity in his life, which introduces another significant privacy concern.

The primary concerns in widespread use of biometrics for remote and onsite authentication are in i) template protection, ii) privacy of the user, iii) trust between user and server, and iv) network security. For civilian applications, these concerns are often more serious than the accuracy of the biometric itself.

The ideal solution to overcoming all the privacy and security concerns would be to apply a strong encryption (say RSA) on the biometric samples as well as the classifier parameters, and carry out all the computations in the encrypted domain. However, the primary goal of a strong encryption algorithm is to destroy any pattern that would be present in the data. We now need to carry out a pattern classification task (identity verification) in the encrypted domain. These two goals are contradictory. In other words, security/privacy and accuracy seems to be opposing objectives. Different secure authentication solutions achieve their goal through a compromise between privacy and accuracy or by making restrictive assumptions on the biometric data.

M. Tistarelli and M.S. Nixon (Eds.): ICB 2009, LNCS 5558, pp. 899–908, 2009.

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. However, such an approach would require an *algebraic homomorphic encryption* scheme [3], which is not known to exist till date. We show that a specific distribution of work between the client (sensor) and the server (authenticator), coupled with a novel randomization scheme can achieve the goal.

Over the years a number of attempts have been made to address the problem of template protection and privacy concerns and despite all efforts, *a template protection scheme with provable security and acceptable recognition performance has thus far remained elusive* [4]. Jain et al. [4] classifies the existing approaches into two groups: *feature transformation-based* and *biometric cryptosystems*. We will look at the two groups, in light of this security-accuracy dilemma. Detailed reviews of the work on template protection can be found in Jain *et al.* [4], Uludag *et al.* [5], and Ratha *et al.* [6].

The first class of approaches that use feature transformation, such as *Salting and Non-invertible Transform* [4] offers security using a transformation seeded by a user specific key. A classifier is designed in the encrypted feature space. Hence, one cannot employ strong encryption here, which necessarily leads a compromise made between security and the performance. Moreover salting based solutions are usually specific to a biometric trait [7,6]. Kong *et al.* do a detailed analysis of the current biohashing based approaches [8], and concludes that the zero EER reported is obtained in carefully set experimental conditions and unrealistic under assumptions from a practical view point.

Biometric cryptosystems use the biometric as a protection for a secret key (Key Binding approach [9]) or to directly generate a secret key (Key Generation approach [10]). The authentication is done using the key, which is unlocked/generated by the biometric. We note that to provide template protection, key is to be unlocked/generated at the client's end. However, this would become a key based authentication scheme, thus losing the non-repudiable nature of biometric authentication. According to Jain *et al.* [4], biometric cryptosystems such as Fuzzy Vault and Fuzzy extractor, in their true form, lack diversity and revocability, and results in performance degradation as the matching is done using error correction schemes. Biometric cryptosystems, along with salting based approaches introduce diversity and revocability in them. However, one can recover the plain biometric from multiple secrets secured using the same key [11].

The approach that is closest to our proposed one is termed *ZeroBio* authentication, proposed by Nagai *et al* [12]. It makes use of client side computation and communication between the client and the server to classify a biometric feature vector using a 3-layer neural network. The client computes the outputs of the hidden layer and transfers it to the server, which completes the authentication by computing the output values of the neural network. The mechanism of zero-knowledge proof using communication is used to ensure honesty. The method is both efficient and generic, however, the server can 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.

This paper proposes an approach that is generic in the sense that we can implement a generic and powerful classifier such as support vector machines(SVM). Moreover, we achieve complete privacy, as the biometric that is passed to the server is encrypted using strong asymmetric encryption. We also achieve efficiency in computation using interaction with the client along with a novel randomization scheme, while maintaining the security of the server templates. In short, we addresses all the concerns mentioned in the introduction. Specifically, $1)$ the use of strong encryption addresses privacy concerns, $2)$ a third party based enrollment scheme takes care of the trust issues. $3)$ provable protection is provided against replay and client-side attacks, $4)$ user tracking is avoided by using different templates for different servers.

The framework provides the ability to classify any feature vector, and hence is applicable to multiple biometrics. Moreover, as the authentication is directly based on the biometric, non-repudiable nature of biometrics is fully utilized. Note that the proposed approach does not fall into any of the categories discussed before, and opens a new direction of research to look at privacy preserving biometric authentication.

2 Authentication in Encrypted Domain

To explain the protocol in a simple setting, we consider the problem of verification as that of classifying genuine and imposter samples using a perceptron. The protocol can be extended to more generic and powerful classifiers, such as *Support Vector Machines* (SVMs). The only restriction to the approach is that the feature vector is assumed to be of fixed length n. Note that even for biometrics such as fingerprints, one can define fixed length feature representations [13].

2.1 Authentication

Let ω be the parameters of the linear classifier, such that the user is accepted if $\omega \cdot x < \tau$, where τ is a threshold. As we do not want to reveal the parameter vector (ω) or the test sample (x) to the server, we need to carry out the computations in the encrypted domain. To achieve this, we use a class of encryptions that are multiplicative homomorphic [14]. 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, which satisfies this property. Note that if we have an encryption scheme that is homomorphic to both addition and multiplication (algebraic homomorphic), we can carry out the computation, directly at the server side. However, such an encryption has not been discovered till date.

During enrollment, the server receives the client's public key, E, as well as the classifier parameters vector ω in the encrypted form, i.e., $E(\omega)$. The authentication happens over two rounds of communication between the client and the server. To perform authentication, the client locks the biometric test sample using her public key and sends the *locked ID* ($E(x_i)$) and the username to the server. 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

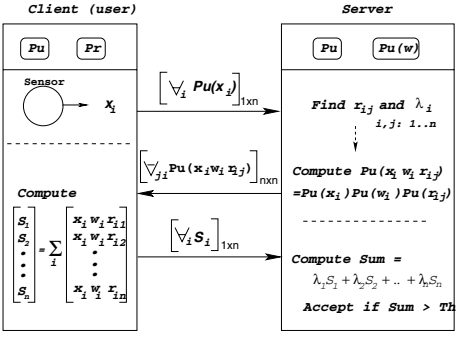


Fig. 1. The proposed authentication process using linear classifier

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.

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:

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

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

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

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

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

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 λ_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, and returns their sum $S_j = \sum_{i=1}^{n} \omega_i \, x_i \, r_{ji}$ to the server
7: Server computes $S = \sum_{j=1}^{k} \lambda_j \, S_j$
8: **if** $S > \tau$ **then**
9: return $Accepted$ to the client
10: **else**
11: return $Rejected$ to the client
12: **end if**

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 4.

As noted before, the linear classification was used for illustration of the algorithm, as it is instructive and easy to understand. The extension of this approach to compute more complex functions such as the kernelized inner products are given in section 3. 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.

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 (as we show in the next section) and hence any verification problem that can be carried out using a generic SVM-based classifier can be modeled by this protocol. One could also extend the protocol to work with the Neural Networks.

3 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. We will now extend the proposed approach to deal with the kernel form of the linear classifier, the support vector machine (SVM).

Kernel-based classification: In the linear case, we described a procedure, $secure-Product$, 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: $S = \sum_{i=1}^{N} \alpha_i d_i \kappa({v_i}^T x) =$

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

We can extend the $secureProduct$ procedure to deal with kernel based classification as well. 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 $secureProduct$. 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. One can solve this problem using another round of communication with the client and define a completely blind kernel-based verification protocol.

Security Extensions: The client end contains the biometric acquisition device as well as keys for encryption and decryption. Security at the client end is critical, especially when using a public terminal to access any service. One could move the private key to a card or carry out the decryption operation, completely in a smart card in case of insecure environments. Another option would be to secure the private keys at the client end using a fuzzy vault [9], which is unlocked only with the biometric that is provided for authentication. This provides a double layer of security through the biometric provided by the user.

RSA is just one of the public key encryption algorithms that is multiplicative homomorphic. This can be replaced by other similar encryptors. One could analyze the computation cost and security issues for each encryption method. A further speed up is possible by reducing the number of support vectors [15].

4 Security, Privacy, and Trust Issues

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. Security is hence a function of the specific biometric used as well as the overall design of the system.

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 provides very little privacy, and hence demands trust between the user and the server. We now take a closer look at the security and privacy aspects of the proposed system.

4.1 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.

Server Security: Let us assume that the 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.

In case the hacker is in the server *during* the authentication, he can try to extract information from his entire "view" of the protocol, i.e. the encrypted classifier parameters $E(\omega_i)$, encrypted test vector $E(x_i)$ and other intermediate data such as random numbers r_{ij}'s, λ's etc. We ask what hacker can learn about the critical data, viz., ω_i's and x_i's? The hacker only obtains k linear equations over the n variables $y_1, y_2, \ldots, y_n$, namely, $S_j = \sum_{i=1}^{n} r_{ji} y_i$ for all $j \in [1, k]$, where $y_i = \omega_i x_i$. Thus, if at all the hacker obtains some information, it is only about the y_i's and not about the ω_i's or x_i's. Notwithstanding, nothing additional is revealed even about the y_i's, by choosing k to be such that $|\mathbb{D}| \geq |\mathbb{Y}|^{\frac{n}{n-k}}$, where $\mathbb{Y}$ is the domain of y_i's and $\mathbb{D}$ is the domain of r_{ji}'s.

Client-End Security: A hacker having access to the user's computer, will not be able to carry out the authentication, as the biometric is not stored on the client's computer. The private key itself is often hard wired into the decryption hardware and often cannot be read out. However, in the worst case, the attacker may be able to decrypt messages in a black-box fashion. Even in such a case, the ω_i's cannot be obtained by the attacker from his entire view of the protocol, as long as $|\mathbb{D}| \geq |\mathbb{Y}|^{\frac{n}{d(k-n)+n}}$, where d is the number of fake authentication requests made to the server. Reconciling this case with the previous one, and taking the limit $d \to \infty$, we get the optimal value of $k = n$.

Network Security: An attacker having control over the insecure network can watch the traffic on the network, as well as modify it. However, all the traffic on the network are encrypted either using the clients public key or using the random numbers generated by the server. Hence 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.

4.2 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. i) 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. ii) 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.

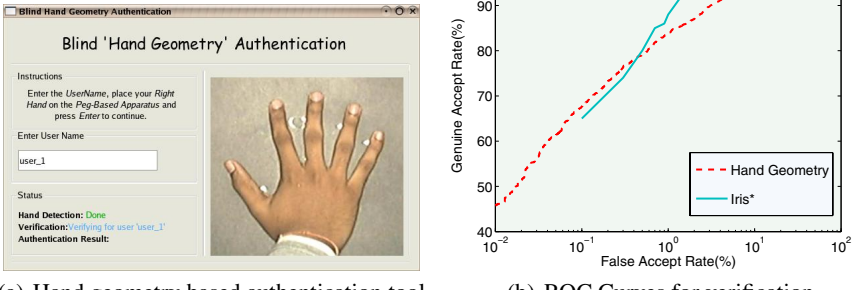
5 Implementation and Analysis

An authentication protocol based on a client-server model was implemented that can perform verification over an insecure channel such as the Internet. The following experiments and analysis evaluates the accuracy and performance of the proposed approach for verification.

5.1 Implementation

For the evaluation purpose a generic SVM based verifier based on a client-server architecture was implemented in GNU/C and the *XySSL* RSA library. All experiments are conducted on AMD X2 Dual Core 4000+ processor, 750MB DDR2 RAM and over the network.

Both exponentiation operations in encryption and decryption assumes that the data consists of positive integers. The feature vectors and the SVM parameters are mapped to integers in the twos complement form after scaling to retain the precision. Thus all our computations are now done in two's complement arithmetic. If x_i is a parameter to be encrypted, the forward mapping is defined as: $x'_i = compl(\lfloor s.x_i + 0.5 \rfloor)$, where s is a scale factor, depending on the range of values for x_is, and $compl()$ maps the integral numbers to its twos complement. The corresponding reverse mapping is done by the server, once the results are obtained. Figure 2(a) shows a hand-geometry based authentication tool that we have implemented based on the proposed method.



(a) Hand geometry based authentication tool (b) ROC Curves for verification

Fig. 2.

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 effectiveness of using SVMs as a classification model for biometric verification problems, we tested it on two different modalities. The verification accuracies after 3-fold cross validation on each of the datasets is presented in Table 1.

The first set of experiments was done on the CASIA IRIS database. The Version 1 of the dataset consists of 108 users with 7 images per user (the seven images are

Table 1. Verification accuracy on biometric datasets

Dataset	# of Features	Avg num of Support Vectors	Accuracy
Hand Geometry	20	310	98.38%
CASIA Iris	9600	127	98.24%

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. For the second set of experiments, we used a hand-geometry dataset that was collected in-house. The dataset 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.* [16]. For each experiment 4 images per user were used for training purpose and the remaining 6 were used for testing. Figure 2(b) shows the *receiver operating characteristic (ROC)* plots for the biometrics using fixed length feature vector representation[1]. The accuracies are similar to those obtained by running SVM in the plain domain, and hence shows the effectiveness of the proposed scheme for biometric based verification problems.

5.2 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 [18], 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$\leq$n, each of which is an integer. 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 needs to generate 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 kernel based classifier with n_v support vectors, one would need to repeat the *secure product* n_v times, once for each support vector. In addition, there is one round of *secure product* to compute 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.

[1] Wang *et al.* [17].

6 Conclusions

The proposed method for biometric authentication is extremely secure under a variety of attacks and can be used with a wide variety of biometric traits. As the verification can be done in real-time with the help of available hardware, the approach is also 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 also enhance the utility of the approach.

References

1. Jain, A.K., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology 14(1), 4–20 (2004)
2. Ratha, N.K., Connell, J.H., Bolle, R.M.: Enhancing security and privacy in biometrics-based authentication systems. IBM Systems Journal 40(3), 614–634 (2001)
3. Fontaine, C., Galand, F.: A survey of homomorphic encryption for nonspecialists. EURASIP J. Inf. Secur. 2007(1), 1–15 (2007)
4. Jain, A.K., Nandakumar, K., Nagar, A.: Biometric template security. EURASIP J. Adv. Signal Process. 8(2), 1–17 (2008)
5. Uludag, U., Pankanti, S., Prabhakar, S., Jain, A.K.: Biometric cryptosystems: Issues and challenges. Proceedings of the IEEE 92(6), 948–960 (2004)
6. Ratha, N., Chikkerur, S., Connell, J., Bolle, R.: Generating cancelable fingerprint templates. IEEE Trans. on PAMI 29(4), 561–572 (2007)
7. Teoh, A., Jin, B., Connie, T., Ngo, D., Ling, C.: Remarks on BioHash and its mathematical foundation. Information Processing Letters 100(4), 145–150 (2006)
8. Kong, A., Cheung, K., Zhang, D., Kamel, M., You, J.: An analysis of biohashing and its variants. Pattern Recognition 39(7), 1359–1368 (2006)
9. Juels, A., Sudan, M.: A fuzzy vault scheme. DCC 38(2), 237–257 (2006)
10. Dodis, Y., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 523–540. Springer, Heidelberg (2004)
11. Walter, J.S., Boult, T.E.: Cracking fuzzy vaults and biometric encryption. In: Biometrics Symposium (2007)
12. Nagai, K., Kikuchi, H., Ogata, W., Nishigaki, M.: ZeroBio: Evaluation and development of asymmetric fingerprint authentication system using oblivious neural network evaluation protocol. In: Proceedings of ARES 2007, pp. 1155–1159 (2007)
13. Farooq, F., Bolle, R.M., Jea, T.Y., Ratha, N.: Anonymous and revocable fingerprint recognition. In: Proceedings of the Biometrics Worshop (CVPR 2007), pp. 1–7 (2007)
14. Menezes, A., van, O., Paul, C., Vanstone, S.A.: Handbook of Applied Cryptography (1996)
15. Abe, S.: Support Vector Machines For Pattern Classification. Springer, Heidelberg (2005)
16. Jain, A.K., Ross, A., Pankanti, S.: A prototype hand geometry-based verification system. In: Proceedings of the AVBPA 1999, pp. 166–171 (1999)
17. Wang, Y., Han, J.: Iris recognition using SVM. In: Yin, F.-L., Wang, J., Guo, C. (eds.) ISNN 2004. LNCS, vol. 3173, pp. 622–628. Springer, Heidelberg (2004)
18. Blum, T., Paar, C.: High-radix montgomery modular exponentiation on reconfigurable hardware. IEEE Transactions on Computers 50(7), 759–764 (2001)