Biometric Template Data Protection in Mobile Device Environment Using XML-database

Rubathas Thirumathyam
Department of Informatics and Mathematical Modelling
Technical University of Denmark

Mohammad O. Derawi
Norwegian Information Security Lab.
Gjøvik University College

Abstract—**This paper proposes a non-traditional XML database which supports biometric templates and provides an API which can be used by independent applications in mobile device environments. Until recently biometric systems are becoming more and more visible in mobile phone devices including fingerprint recognition or gait recognition. To gain a real understanding of how it is possible to protect the biometric data, this paper first starts out with introducing a technique for security in a biometric system and emphasizes that template protection is important by going through the vulnerabilities and threats. Further, it points out requirements for template protection, recital of various template protection schemes and a brief overview of biometric standards.**

I. Introduction

Biometric systems have evolved into a security arrangement, that can be trusted to authenticate users' access to a given system. One of the advantages by using a biometric security system is that the users can avoid remembering a password or any other secret combination to be authenticated and gain access to a system. Another (theoretical) advantage is that a truly biometric identifier is only available to the subject itself and can never be forged. These advantages should ensure a more secure system and also enhance the user experience in the authentication process.

An article [1] proposes biometric gait recognition using mobile devices. The research team has already implemented one Java application on an iPhone-based phone containing an embedded accelerometer sensor. With the Java application, they have completed preliminary data analysis and obtained results on using gait to determine the correct identity of a given person walking. Furthermore, it shows that it is possible to successfully implement biometric gait recognition on a mobile device. Nevertheless, the article does not mention a security analysis of the system and a system is only as secure as its weakest link. The problem is if the mobile device gets stolen, it will become useless, since the thief would most probably seek to try to gain access by attacking in different kind of ways than trying to replicate the gait of the original owner.

Every security system is exposed to attacks and one crucial attack, in our case, is on the biometric system database. Most biometric systems authenticate a user based on a local biometric *template* ($\mathcal{T}$) and a biometric *sample* ($\mathcal{B}$) given by the user during the authentication process. Obviously, if the protection of the biometric template is not optimal it can lead to successful attacks on the biometric system and eventually to unauthorized access. Securing data on mobile embedded devices is hard and not direct forward, since there exist limitations (speed, memory, battery and changing environments).

In this work we have not conducted any experiments to prevent an attack on the templates, instead we illustrate a research proposal on how it is possible to secure the data.

II. Biometric system

The conceptual architecture of a biometric system consists of five major components [2]: *sensor*, *feature extractor*, *template database*, *matcher* and a *decision* component[1]. Furthermore, the system can in a given moment be in one of two phases. One phase is *enrollment* and the other is *authentication*.

A. Enrollment

In the enrollment phase, the system registers, with the aid of a sensor, a user's behavior or physiology - earlier referred to as a biometric sample. From the registered sample, biometric *features* are extracted with the help of a feature extractor. These features will be a subset of data from the original sample, this reduction in data can remove any superfluous information, which is not relevant in the later authentication phase. Furthermore, to only save relevant data will improve the resource and speed performance of the overall system. In a security system it is also a good paradigm to save only the least information as possible due to privacy concerns. These features are saved in the biometric template. However, human behavior is not always deterministic, and because of variations in the actual measurement of the behavior or physiology, it can be an advantage to register multiple samples from the user in the enrollment phase. Finally, the generated template is saved in the biometric system database.

B. Authentication

In the authentication phase, the user gives a 'live' biometric sample. This sample is then compared with a biometric template from the biometric system database. The template can either be retrieved by claiming an identity or letting the sample be compared to all existing templates in the database. This comparison is made in the matcher component and would output a *match score*. The identity decision is, however, made

[1]In some systems the matcher and decision components are joined together

by the decision component and will, based on the decision, either authorize or deny access to the system.

C. Vulnerabilities

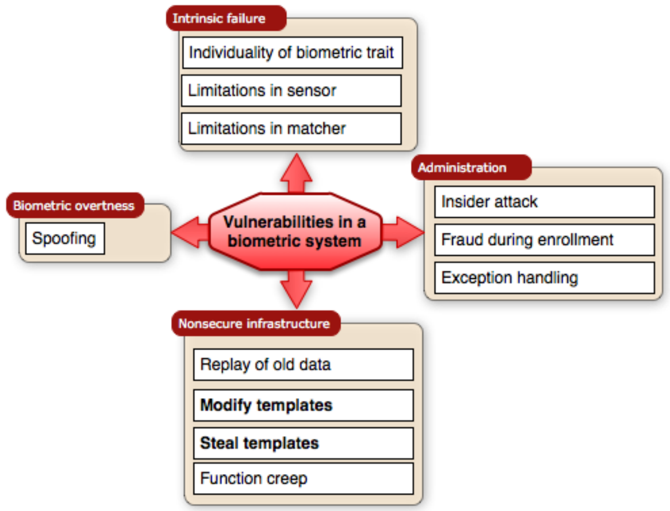

Fig. 1. Various vulnerabilities in a biometric system [2]

There exist different biometric system vulnerabilities and the causes [3] are adapted from the fish-bone model [2] in figure 1 . Exploiting these vulnerabilities can lead to severe attacks on the biometric system. [4] identifies eight possible attacks in a biometric system.

Furthermore, [2] distinguishes between *intrinsic failures* and *failures due to an adversary attack*. Intrinsic failures occur when an internal error occurs in one of the five components. Especially two failures are of utmost importance: *false accept* and *false reject*[2].

False acceptance is the case where either an attacker tries to gain access with a forged biometric sample and the system authorizes the attacker or if a user is accepted as a different user. False rejection occurs when an authentic user is denied access based on a given biometric sample. Both rejections is a result of the algorithm for feature extraction being too vague, thereby the sampling of the user's features is not sufficiently distinguishable and the system cannot (correctly) single out the correct template. If the biometric sample even cannot be measured (e.g., malfunctioning hardware or too much noise) failures such as *failure-to-enroll* (FTE) or *failure-to-acquire* (FTA) arises.

In addition, variations in behavior and noise in the sensor measurement can lead to inaccurate biometric samples which again can affect the decision of authorization. To ensure reliability it is necessary to control and calibrate the technology used for collecting the biometric sample. [5] describes progressing research within design of sensors, algorithms and template schemes to reduce the probability of intrinsic failures; it also claims advantages in making use of fusion methodologies to integrate multiple different biometric features, called *multibiometrics*.

1) Biometric template database: In this proposal we will emphasize protecting the template, thus it the template database (in figure 2) which is relevant. An attack on the biometric template database is possibly the most critical in the whole biometric system, which can lead to these three vulnerabilities [2]:

- Replacement of a forged or invalid template
- Reconstruction of biometric samples from template
- Abuse of the templates to cross-match with other applications

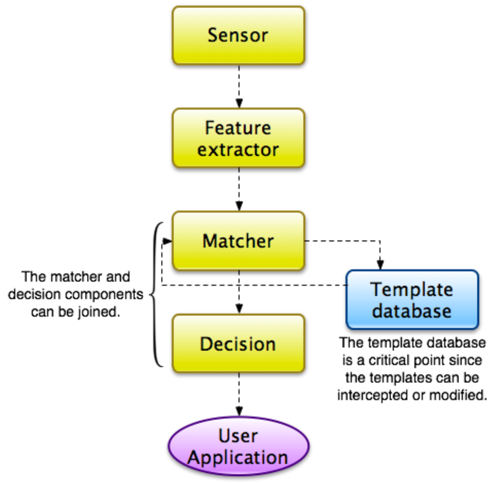
Sensor
Feature extractor
Matcher
Template database
Decision
User Application
The matcher and decision components can be joined.
The template database is a critical point since the templates can be intercepted or modified.

Fig. 2. Template database is a critical point in a biometric system, adapted from [2]

D. Privacy threats

A person has limited biometric identifiers and ironically some of their strengths are also their weaknesses. There are multiple privacy threats regarding a biometric identifier. One is that a biometric identifier represents a real identity and thereby always can be linked to a real life person. [6] mentions that it is possible to extract critical information from a saved biometric template. Many biometric features reveals ethnic background, genetic or medical conditions. Thus, speculative companies can abuse this information, e.g., to calculate the probability for feature health conditions for a given person and estimate an insurance premium based on that[3].

Moreover, biometrics cannot be updated or revoked. If a biometric system is compromised and an attacker gets hold of the biometric features or templates it can cause a breach in security and privacy. [7] has proved that a full fingerprint can be reconstructed from the minutia points. Similar reconstructions could be done with other biometrics. Thus, the attacker has the opportunity to impersonate the users registered in the biometric system.

[2]Derivatives are False Accept Rate (FAR) and False Reject Rate (FRR), which are statistical quantities that can indicate the accuracy performance

[3]Just an example - a discussion in ethics is not in this proposal's scope

III. Requirements for protection

Based on the aforementioned vulnerabilities and threats, all biometric templates in biometric system should be optimally secured and protected. Nevertheless, [6] puts up a set of requirements for template protection which safeguards both privacy and security in biometric systems.

A. Privacy requirements

- **Identity privacy:** The relation between the biometric data and identity data must be highly protected. So if either one is revealed it would still be difficult to track down the other.
- **Irreversibility:** Transform the biometric sample into a biometric template which cannot be transformed back to the sample again. In case templates are leaked it should not be possible to reconstruct the original biometric features.
- **Unlinkability:** Biometric template used in one application for a given user and a given set of biometric features should not be identical or similar in other applications using the same algorithm for generation of biometric templates. Making it hard for the attacker to determine if 'different' templates are identical or generated by the same algorithm.

B. Security requirements

- **Confidentiality:** Both the biometric database and all communication between each component should be protected against all trivial *man-in-the-middle* attacks and modification of data.
- **Integrity:** Since biometric features are highly volatile, accuracy and integrity of data are important. All components should be reliable in both phases of the biometric system.
- **Renewability and revocability:** There exist a high desire to make use of templates which either can be renewed or revoked. Otherwise, a user can never re-access a biometric system once his or her biometric template is compromised or stolen. A possibility is to be able to extract a different set of biometric features from the sample to generate a new biometric template[4].

IV. Template protection schemes

The requirements are meant to help create a template protection scheme. Nevertheless, the protection should not devalue the system performance (FAR/FRR or speed). The template protection schemes we have looked at can bluntly be split into two categories [2]: *Feature transformation* or *Biometric cryptosystems*.

[4]Provided that the attacker was not able to reconstruct the complete biometric sample from the template

A. Feature transformation

A feature transformation scheme applies a transformation function ($\mathcal{F}$), to the biometric template. The function, however, often requires parameters which are obtained from a random key ($\mathcal{K}$) and thereby creating a transformed template ($\mathcal{F}(\mathcal{T},\mathcal{K})$). Because it is only the transformed template that is saved in the database, this transformation happens in both enrollment and authentication phase.

1) Salting: The salting transformation function uses a key or password to create the transformed template, but this template is invertible, so it make demands to protect the key or password.

2) Non-invertible transform: A non-invertible transformation makes use of a *one-way* function, so the transformed template will be easy and quick to generate - but trying to find the given original template as input will computationally be hard.

B. Biometric cryptosystems

Biometric cryptosystems deviate from feature transformations by making use of external data, *helper data* ($\mathcal{H}$). Helper data does not contain many details about the original biometric template.

1) Key-binding biometric cryptosystem: When a biometric cryptosystem is key-binding the (key in the) helper data is independent of the biometric template. The helper data is created by using both the template and a key. This key is obtained from a sub application (which generates a cryptographic key).

2) Key generation biometric cryptosystem: Opposite of being independent of the biometric template, a key generation biometric cryptosystem makes use of the biometric template in the generation of the key.

C. Summing up

In [2] a comparison between the four schemes have been made and their conclusion is, that there exist no 'best' approach for template protection. A biometric system's purpose and scenario has a huge impact on which scheme is suitable. Nevertheless, within each scheme there is research going on to optimize known problems in the matcher, basically so the *inter-* and *intra-user* variations are better handled. Other research is going on to clear out the requirements of the schemes. [8] indicates that some of the security requirements often are misunderstood and claims the common (practical) techniques are lacking. Hence, an optimal scheme for the biometric system used in [1] has to be carefully designed based on an analysis of the different scenarios involved of the application.

V. Biometric standards

The biometric industry has existed for many years. Helped by horrible terrorist events and criminal attacks there has been a focus on creating a biometric standard to prevent impersonation and theft of identities. Particularly, a lot of effort has been put in specifying a standard that enhances the interoperability between application domains. Even though biometric security systems become more and more mature, biometric is a vast concept and there is a need for an open standard. In the literature their exist ambiguous and 'private' terminologies [6]. A standard will help specifying proved and thought-through security and privacy requirements. Especially, seen from a distributed and *interoperable* point of view it is crucial to have standards, so different and otherwise distinct biometric systems can interoperate. The (later described) BioAPI is an example of a closed standard, but open standards should be considered and an open standard does not necessarily mean that the system becomes less secure, in best case quite the contrary [9].

A. BioAPI

One of the first key specifications was the *Biometric Application Programming Interface* (BioAPI). BioAPI specifies how different applications can interact with each other and make use of high-level generic biometric enrollment and authentication. It even includes a database interface and enables various applications to have different *roles*. So a client-application can capture the biometric sample, transmit it to a server-application - which does the actual processing and authentication - and respond back to the client-application.

B. CBEFF

Nonetheless, to support interoperability of biometric data and to do so platform independently (which was one of the requirements of BioAPI) there was a need for a *data description format*. So the BioAPI consortium members came up with the *Common Biometric Exchange File Format* (CBEFF). CBEFF was made to represent different kinds of biometric data, ranging from advanced biometric features, which required complex data structures, to simplistic ones. This flexibility gave the designer of a biometric system the opportunity to define biometric data structures according to the platform and hardware the biometric system will be running on.

C. XCBF

One of the two formats defined in CBEFF made use of *Abstract Syntax Notation One* (ASN.1). Even with a standardized description language in CBEFF, the format was binary-based and could not overcome the challenges in data transmission on the Internet [10], where XML had gained foothold. So it cleared the way for a new specification. XML had become more and more the common format for data transport over the internet, especially in the hype of *Web 2.0*. So defining a biometric XML schema, XML Common Biometric Format (XCBF) [11], was applauded.

VI. Proposed XML Databases

A lot of data is extracted from traditional databases and converted to XML documents [12], hence it may be more efficient to store the data in XML, instead of creating the conversion overhead for every request. Thus, many *traditional databases*[5] began to support XML [13], however, they typically implemented an extra 'XML-layer' on top of the existing database management system. Thereafter the conversion was just moved from elsewhere to the database, not reducing the overhead significantly. The term *XML-enabled* was given to these kinds of databases. There emerged a different kind of database, *non-traditional databases*[6], which were *XML-native*. Opposed to a traditional database, which will have tuples as its fundamental unit of logical storage, a native XML database will use an XML document as its fundamental unit of logical storage.

A. BaseX

BaseX is an example of a native XML database. This specific database supports the two APIs, *XML:DB API* (XAPI) and *XQuery API for Java specification* (XQJ). Both APIs try to help create queries in an easy manner, the latter with the use of W3C XQuery specification and supporting *ACID-safe transactions*[7]. The database seems to be lightweight, yet powerful and efficient to be installed on a mobile device.

B. A biometric XML Database

As of today there exist no (XML) database, which supports the XCBF specification. Nevertheless, there is an indigence of a secure storage system that can protect the biometric templates needed in a biometric system. As non-traditional (text) databases becomes more and more favored on embedded and mobile devices, it makes sense to take advantage of a biometric XML schema to save the biometric templates.

VII. Proposed Implementation of XCBF:API

It would be convenient to extend one of the existing native XML databases, which does a good job of providing basic features and API support. BaseX is written in Java and can easily be deployed to the iPhone's, so extending BaseX with the XCBF specification would be one practical approach.

A. XCBF:API

A biometric (database) API, *XCBF:API*, gives applications on the embedded, mobile device a way of communicating with a central biometric template storage, hence the opportunity to interoperate with each other. Creating a Java library/package (e.g. `org.basex.xcbf`) with classes and interfaces that provides an API, following the XCBF standard, is straight forward.

One of the key features of the new API is to provide methods to create a biometric template (based on the XCBF schema) and thereafter verify and validate the XML document (biometric template) according to the XCBF schema, when the application tries to store it into the template database. There exist multiple tools and technologies to help with this - *Java API for XML Processing* (JAXP) would be ideal to make use of.

[5]Often equal to *Relational Database Management Systems* (RDBMS)

[6]These databases distinguish themselves by being specifically optimal in handling one kind of data - e.g., text, spatial or temporal

[7]http://basex.org/

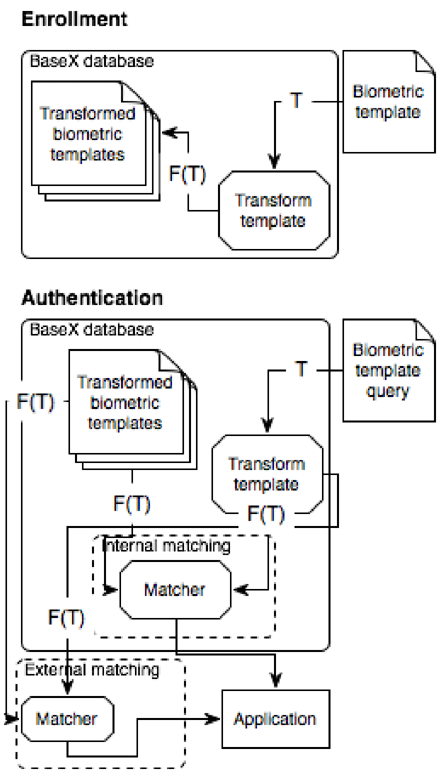

Fig. 3. Enrollment and authentication phases after implementing XCBF with feature transformation in BaseX

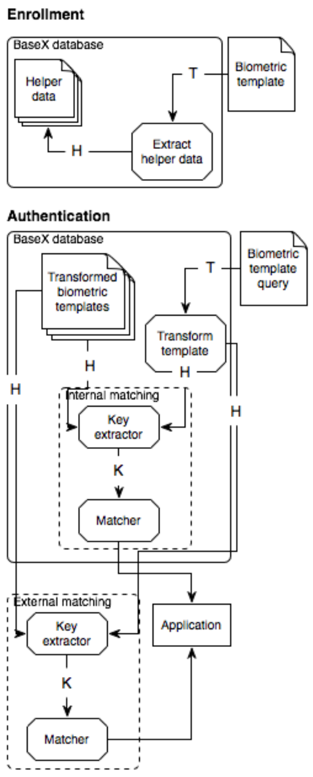

Fig. 4. Enrollment and authentication phases after implementing XCBF with a biometric cryptosystem in BaseX

Beside implementing the XCBF specification into an API, it is an advantage to extend the API further with functionalities, that makes the database attractive for applications who intend to make use of it as a part of a biometric security system. The following extensions are complicated and should be designed and modelled carefully, with especially the earlier stated requirements in mind.

B. Build-in template protection

Since XCFB is a specification for the data format it does not provide the actual template protection schemes. Instead of each application using its own template protection scheme it would be greater to integrate multiple standard template protection schemes in the database. Via the API each application can make use of whichever protection scheme that suits their application.

A prerequisite for the extended API is to implement the template protection schemes and fulfill the security and privacy requirements stated in section III. Figure 3 shows an illustration of how a template gets transformed using feature transformation during enrollment and authentication. In the authentication phase there can either be an internal or external matcher. An application making use of an external matcher has to query with a biometric template and the database will return a transformed template, which can be matched with a transformed template got with a claimed identity from the database. Letting the database provide enhanced template protection for a biometric template.

C. Internal matching

Furthermore, we propose to join the matcher and storage component into **one** biometric template protection database. this is illustrated in the figure with the internal matching, so that the database in fact can do the matching of the templates and output a *matching score*. This score can then be used by an application to make a final decision.

Figure 4 shows how the biometric template can get stored using a biometric cryptosystem. As earlier stated the database can either store a template which is bound with a cryptographic key independent of the template or one where the cryptographic key is generated based on the template. The matching is done by checking the validity of the extracted key, which again can be done either internally or externally. This type of system should also take into consideration the intra-user variations in the biometric template - e.g. to impose *error correction* mechanisms.

D. Hybrid template protection schemes

The proposed template protection schemes in the current literature are either vague or too simplistic [2]. As earlier stated an optimal template protection scheme for one application, is maybe not suitable for another application or even in another context of the same application. We propose to study these schemes further with regard to the possible user scenarios and design hybrid schemes which can transform multiple (different) biometric samples or template. So the overall biometric system becomes truly multi-biometric. Whichever application who wants to make use of the database can choose the optimal protection scheme, without being troubled to implement one by themselves.

E. Interoperability

Another incentive - to deploy a biometric system with this API and database - is to allow different applications to interact with a given user's template. Basically, the enrollment process can happen once and all applications can make use of the saved biometric template and correctly authenticate a user, since there (in theory) only can exist one identity with the given biometric template.

F. Implementation

Each of the extensions is going to be a part of the database and has to be somehow modelled and thereafter implemented. The actual transformations of a template into a transformed template could be done with *XSL Transformation* (XSLT), which is a declarative powerful XML-based language. XSLT's core purpose is to transform one XML document into another XML document based on template rules from a *XLTS-stylesheet*. Figure 5 shows how the flow works for transforming one XML document to another, where the resulting document would be the transformed biometric template (or helper data) and the processor can be achieved with the help of JAXB.

Doing the internal matching (especially intrauser variations), hybrid template protection schemes and interoperability is partly to model algorithms, heuristics and implement them using various XML-based languages (*XQuery* and *XPath*) there are to manipulate or query a document. So these various 'scripts' will run in the backend of the database, while the API (frontend) will provide elegant embedded functions and methods.

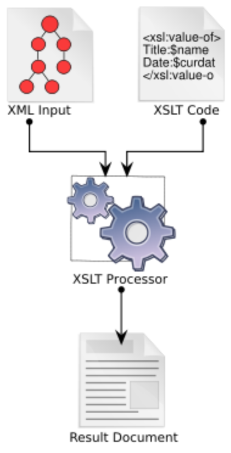

Fig. 5. Showing the transformation flow [14]

VIII. Future work

The XCBF:API should lead to an efficient, biometric secure and (not only non-traditional, but) newfangled database. The research proposed would lead to hybrid template protection schemes and a biometric system capable of handling multiple biometric features. The application will be able to distinguish between different varieties of schemes, choosing the one which is most suitable. The rapidly growing market for mobile devices with sensors that can extract various biometric features needs a database, that provides the necessary protection, security, reliability and interoperability.

The general threat model for storing templates in a database can significantly be eliminated if the XCBF:API is implemented in a database.

IX. Conclusions

In this proposal we have given an overview of the conceptual architecture of a biometric system and how template protection is an essential part of such a system in mobile device environments. The XCBF:API is put forward and proposed as a protection to secure templates in a database. This API would protect biometric templates efficiently on mobile devices, since a customizable template protection scheme can be chosen. We favour a non-traditional (XML) text database over a traditional due to the limitations on a mobile and embedded device. Furthermore, an XML database provides us with querying and data manipulation (e.g, XQuery and XPath) facilities, which can be more abstract and suitable than resource-demanding (low-level) SQL. Finally, we prosed a database which is application and technology independent. Thus, a high level of interoperability between various applications can be achieved.

References

[1] M. Tamviruzzaman, S. I. Ahamed, C. S. Hasan, and C. O'brien, "epet: when cellular phone learns to recognize its owner," in *SafeConfig '09: Proceedings of the 2nd ACM workshop on Assurable and usable security configuration*. New York, NY, USA: ACM, 2009, pp. 13–18.

[2] A. K. Jain, K. Nandakumar, and A. Nagar, "Biometric template security," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. Article ID 579416, December 2007.

[3] A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: a tool for information security," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 125–143, 2006.

[4] N. K. Ratha, J. H. Connell, and R. M. Bolle, "An analysis of minutiae matching strength," *Audio- and Video-Based Biometric Person Authentication*, vol. LNCS 2091, pp. 223–228, 2001.

[5] A. Ross, K. Nandakumar, and A. K. Jain, *Handbook of Multibiometrics*. Berlin: Springer, 2006.

[6] J. Breebaart, B. Yang, I. Buhan-Dulman, and C. Busch, "Biometric template protection - the need for open standards," *Datenschutz and Datensicherheit*, vol. 5, pp. 299–304, 2009.

[7] A. Ross, J. Shah, and A. K. Jain, "From template to image: reconstructing fingerprints from minutiae points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 544–560, April 2007.

[8] L. Ballard, S. Kamara, and M. K. Reiter, "The practical subtleties of biometric key generation," *USENIX Association*, 2008.

[9] C. P. Pfleeger and S. L. Pfleeger, *Security in Computing*. New Jersey: Prentice Hall, 2007.

[10] B. Wirtz, "Biometric standardisation," *Biometric Technology Today*, vol. 10, no. 8, pp. 8–11, September 2002.

[11] T. Aichelen and et al, *XML Common Biometric Format*, 2003, oASIS Standard.

[12] S. O'Connell, *Advanced Databases Course Notes*. Southampton: University of Southampton, 2005.

[13] J. Melton and S. Buxton, *Querying XML - XQuery, XPath, and SQL/XML in context*. San Francisco, CA: Morgan Kaufmann, 2006.

[14] "Xslt (xsl transformations)," http://en.wikipedia.org/wiki/XSLT.