Automated Forensic Fingerprint Analysis: A Novel Generic Process Model and Container Format

Tobias Kiertscher[1], Claus Vielhauer[1,2], Marcus Leich[2]

[1] University of Applied Sciences Brandenburg, Dept. of Informatics and Media, PO Box 2132, 14737 Brandenburg a. d. Havel, Germany
[2] Otto-von-Guericke University of Magdeburg, Dept. of Computer Science, Research Group Multimedia and Security, PO Box 4120, 39016 Magdeburg, Germany
{kiertscher,vielhauer}@fh-brandenburg.de, mleich@st.ovgu.de

Abstract. The automated forensic analysis of latent fingerprints poses a new challenge. While for the pattern recognition aspects involved, the required processing steps can be related to fingerprint biometrics, the common biometric model needs to be extended to face the variety of characteristics of different surfaces and image qualities and to keep the chain of custody. Therefore, we introduce a framework for automated forensic analysis of latent fingerprints. The framework consists of a generic process model for multi-branched process graphs w.r.t. security aspects like integrity, authenticity and confidentiality. It specifies a meta-model to store all necessary data and operations in the process, while keeping the chain of custody. In addition, a concept for a technical implementation of the meta-model is given, to build a container format, which suits the needs of an automated forensic analysis in research and application.

Keywords: Automated forensic analysis, automated dactyloscopy, multi-branched biometric process, forensic container format

1 Introduction

Dactyloscopy, i.e. the recovery of latent fingerprints with a wide range of analysis methods for different kinds of surfaces is a commonly used criminal investigation technique. Currently, the recovery is a manual process, done by specialists. However, the manual processing of latent fingerprints is time-consuming and implies physical and chemical modifications of the original trace, due to vaporization, for example. Our research project [1] prospects a contactless way to recover latent fingerprints, with an optical high resolution surface scanner, which produces *topography maps*. One goal of the project is to define an automatic forensic analysis process to recover the fingerprints from a wide range of surfaces. This poses numerous research problems for example in the domains of signal processing and pattern recognition, in order to minimize detection errors. However, in this domain, an additional requirement for handling the forensic process chain has evolved, which will be addressed in this paper.

Because of the varying requirements for the pre-processing and feature extraction algorithms w.r.t. the different surfaces, our pattern recognition process is based on the components of a generic biometric process and embodies a *multi-branched process graph* with a number of alternative and parallel processing steps. In that multi-branched process graph it is possible to select certain algorithms based on the results of other algorithms, just like a dactyloscopy specialist selects certain methods based on his experience. And it is also possible to apply more than one algorithm in parallel and compare the results to select the best.

Integrity and *authenticity* as commonly known in IT security [2], are two important security aspects in that process graph, which is the foundation for our forensic data processing application. To assure the *chain of custody* [3], while processing the topography maps, the integrity and authenticity of the raw data from the scanner needs to be secured likewise the integrity and authenticity of intermediate and resulting data. Furthermore, a set of every algorithm and their *parameter sets*, applied in a *processing step*, needs to be kept as well to follow the Daubert standard [4], which is a set of requirements for legal relevant evidence. Since the potentially recovered fingerprints are highly sensible w.r.t. *privacy*, encryption is another important requirement, when the data is stored or transmitted.

To facilitate the chain of custody and the desired confidentiality in such an automated dactyloscopy process environment, a *forensic data container format* with sufficient support for a multi-branched process graph and the before mentioned security aspects is required. Because we have not been able to identify an existing container format which suits our needs, in our reviews, we propose a novel container format to store an evidence chain in an automated multi-branched dactyloscopy process.

This paper is organized as follows: In the first section of this paper, an introduction is given. In the second section we develop further requirements for the container format, in consideration of the multi-branched dactyloscopy process. We then review the state of the art to evaluate already introduced formats in context of the process. Following the state of the art, we describe the novel container format. Therefore, we introduce a generic process model for an automated forensic analysis of biometric trace in section 4, including a meta-model, which supports the definition of a container format to store the data and operations in each process step, while keeping the chain of custody. (The meta-model describes the logical elements without suggesting any implementation techniques.) Furthermore, we show a way to implement the meta-model with a specific forensic process in section 5. First insights in our technical implementation of the container format are given in section 6, followed by conclusion and presentation of future work in section 7.

2 Requirements of an Automated Forensic Process

As shown in the introduction, one goal of our project is to define an automated forensic analysis process for latent fingerprints. To reach this goal, we apply concepts from the field of dactyloscopy as well as from fingerprint biometrics. Dactyloscopy and fingerprint biometrics are conceptually similar w.r.t. the basic data processing steps

(pre-processing, feature extraction, classification). However, there are additional requirements for dactyloscopy. The chain of custody needs to be preserved in every processing step. And due to the huge variations w.r.t. condition and environment of latent fingerprints, a broad range of algorithms for pre-processing and feature extraction is needed, and must be applied in an adaptive way. As a bridge between these two fields a forensic data container format with the capability to store the data (source, intermediate and result), algorithms and parameter sets of a multi-branched forensic fingerprint analysis process is desirable.

With respect to the chain of custody, the input and output data of each process step can be comprehended as a forensic record with a set of predecessors and an assigned processing algorithm using a parameter set. As a result the container format must support the storage of dependencies between the data, to form a multi-branched graph. The individual process steps can be distributed over network and it is necessary to be able to interrupt the process between process steps, continue later, and still keep the chain of custody. Therefore, the container format must assure integrity and authenticity of every source, intermediate and resulting data, along with the applied processing algorithms. A goal is, according to the Daubert standard, to enable the reproducibility of the whole dactyloscopy process, with the information stored in the container. In addition, the container format needs to support encryption for the data w.r.t. privacy issues.

An important additional requirement for our research is, to have easy access to the data with a broad range of programming languages and scientific tools, e.g. GNU Octave [5]. It is desirable to get at least read access to the data in the forensic container through commonly used software techniques without the need of a programming library, which restricts the retrieval of the data to a certain programming language.

3 State of the Art

The following state of the art, which is relevant to automated analysis of latent fingerprints, discusses the process model and the container format.

Process Model

Amongst the variety, a common process model for a biometric process was introduced in [6] and adopted e.g. by [7]. This process model, shown in Fig. 1,

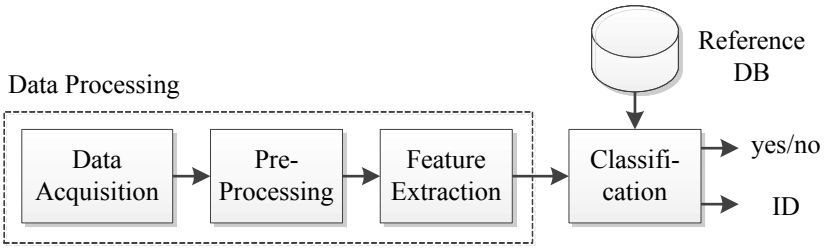

Fig. 1. A general biometric process

includes the three data processing steps “data acquisition”, “pre-processing” and “feature extraction” for the biometric data and a classification stage, which uses a reference database and delivers a Boolean or an ID. This and other models have proven in a large number of biometric applications.

However, in contrast to the controlled environment in most biometric applications, forensic traces, like latent fingerprints, do have much more varying quality and characteristics. To encounter these difficulties, a process for automated dactyloscopy needs to be modular and adaptive and must support a multi-branched process graph. The whole data processing in biometrics is typically done in a trusted environment and there is no such strong need for assuring a gapless chain of custody, as with a forensic analysis in dactyloscopy, which often is distributed over a number of institutes and specialists. Therefore, we are looking for a process model to be represented by a data container format to cover the before mentioned needs.

Container Format

There are commonly used data formats in both fields, forensics and biometrics. In the field of forensics, formats, also called DEC (Digital Evidence Container) or DEB (Digital Evidence Bag), have been introduced, like the AFF (Advanced Forensics Framework) [8], and the proprietary EWF (Expert Witness Format) [9]. Both formats are specialized in storing disk images. AFF supports cryptographic methods for assuring integrity and authenticity, but since it only supports the storage of one disk image and some meta-data in a key-value manner, it is not sufficient for the needs of an automated forensic analysis of latent fingerprints. The EWF is a proprietary format with no public documentation, and is consequently not a good choice for applications in research.

An extended version of the AFF, called AFF4 and proposed in [10], is very flexible and does support multiple evidence data streams, meta-data as simple RDF triples and external references. The AFF4 is designed to support huge evidence corpuses, even if they are stored in a distributed environment. However, since AFF4 addresses the intense use of disk images in a various ways, e.g. remapping for memory analysis, it adds complexity to the format, which is not necessary in the context of a biometric process. Another problem with AFF4 is that, even though the use of hashes and signatures for data elements is possible, it is not mandatory. And there is no way specified, to put a signature of a volume into the volume to encapsulate data for verifying its own integrity.

In the field of biometrics some formats like CBEFF (Common Biometric Exchange Formats Framework) [11], developed by the NIST, and the related XCBF (XML Common Biometrical Format) [12] standardized by OASIS, have been suggested. These formats are designed to be used in biometrical applications and do support integrity and authenticity as well as confidentiality. They are using a single-file concept and their main purpose is the storage of one or multiple biometric data samples, in terms of raw data or biometric templates in a standardized format like the formats specified in [13]. This approach however, will result in a scenario with large raw data blocks (e.g. topography maps) and a couple of intermediate data with comparable

size, in very large files. They do not support compression and since XCBF stores binary data with Base64 encoding, this problem increases even more.

4 Generic Process Model and Meta-Model

As apparently the classic biometric process model, likewise the existing forensic and biometric file formats, do not fit the needs of an automated forensic fingerprint analysis in research and application with support for a chain of custody, we propose a generic process model along with a novel forensic container format with sufficient support for such a process.

4.1 Generic Process Model

Our generalization of the biometric process starts at the data acquisition level, by abstraction to the *initialization* of the process, because a forensic analysis can start with data acquisition, as well as with existing intermediate data. Secondly, the two steps "pre-processing" and "feature extraction" are abstracted to *transformations*, since in a multi-branched process graph for automated dactyloscopy, there are more than two fixed processing steps, which can be applied on demand. The resulting data of a step, including the used parameter set for the acquisition or transformation, is called *entity*. Every process step depends on a *provenance*: The initialization depends on a data *source* and a transformation depends on an *algorithm*. A transformation can use at least one existing entity as input and produces exactly one entity as output. The process can only be interrupted after a step is completed.

To support the chain of custody, every step ends with the signing of the resulting entity. Fig. 2 gives a brief overview to the generic process model. The classification stage of the process is not a subject of this model.

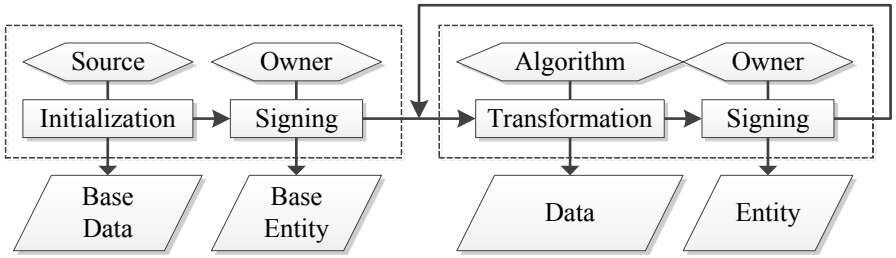

Fig. 2. Initialization and transformation in the generic process

Our assumption in this model is, that the production, transformation and signing of entities is performed in trusted environments (illustrated by dotted boxes in Fig. 2), while the container can be exchanged over untrusted channels.

4.2 Meta-Model

The meta-model defines elements and their relations which are suitable to store the entities of the model introduced in 4.1, and is designed to assure integrity, authenticity and optional confidentiality.

4.2.1 Structure

The elements of the container structure are forming a tree, with the *container* element as the root. The container element is parent of a number of *editions*, one current edition and a stack of past editions. An edition contains a global unique identifier, a timestamp, the description of an authority, called *owner*, which has created or extended the container, and a list with all entities added to the container, while creation and extension, respectively. The stack of past editions is forming the *history* of the container. During the initialization of a container, the current edition is created and the history stack is empty. When the container is extended, the former current edition is wrapped into a *history item*, pushed on to the stack and a new current edition is created. Besides the current edition and the history, an *entity index*, with references to all entities stored in the container, is child of the container element too.

An entity element is the parent of an *entity header*, optionally a *parameter set* from the provenance of the entity, which can be the source or the transformation algorithm, and at least one *value*. The entity header contains a container-wide unique *id*, a list of the id's of all predecessor entities and a reference to the provenance of the entity. Furthermore the entity header references an *entity type*. The definition of the entity type describes all required and optional values of the entity along with their data types. Fig. 3 gives an overview to the meta-model.

4.2.2 Security

To assure the integrity and authenticity of the data, the provenance parameter set and all values of an entity are protected by a signature (*value signature*) as well as the whole entity (*entity signature*). In the following figures, signatures are visualized by the symbol  . At the root of the structure, a *master signature* secures the whole container. Every time the container is extended, i.e. insertion of one or more new signed entities, the former master signature is integrated into the new history item and pushed on to the history stack. In the figures, a past signature is annotated by the symbol  . After the extension, a new master signature is created and integrated into the container element. All signatures, from the value signatures over the entity signatures up to the master signature, are implementing a hierarchical hash tree in analogy to a Merkle-Hash-Tree [14]. However, there is a difference in that every level in the tree not only uses a cryptographic hash, but uses a complete signature as well. As a result the integrity of a partial data structure, like an entity as part of the container, can be verified as well as its authenticity, even if other elements or the root in the structure are corrupted.

The hash of an entity signature does not cover the data of the child elements, like the provenance parameter set or the data of the values, but does only cover the entity header and the signatures of the child elements. The same restriction applies to the master signature. The hash of the master signature covers all child elements of the container element, including the entity signatures, but excluding the data of the entities. This way, the structure of the container can be verified, even if some child elements are corrupted. If, e.g. an entity is corrupted, the master signature is not

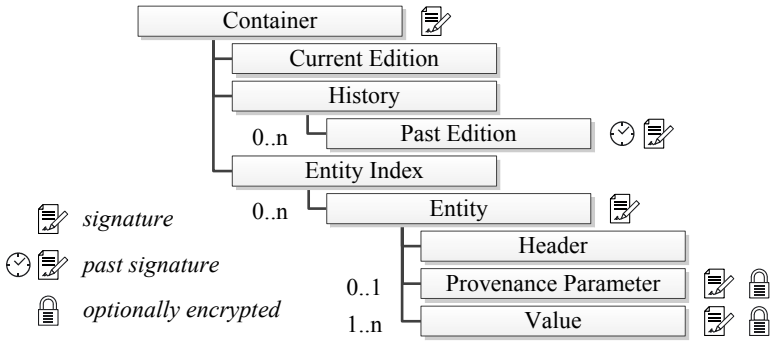

Fig. 3. Overview to the meta-model

affected directly and can still verify the history of the container and the content of the entity index. If the entity signature, covered by the master signature, is checked, it proofs the content of the entity to be corrupted. As a consequence, to proof the integrity and authenticity of a whole container, at first its master signature needs to be verified. After the master signature proofs to be correct, every entity signature needs to be verified as well. And after an entity signature proofs to be correct, the signatures of the entity's children must be verified as well.

To allow confidentiality during transmission of the container, only the parameter sets of the entity provenances and all intermediate data, in terms of entity values, need to be encrypted. Encrypted values must be decrypted prior to any transformation. If continuous confidentiality is required, resulting entity values need to be encrypted also. Respecting that the provenance parameter set and the entity values can be stored in an encrypted manner, meta-data, to describe the encryption, can be included in the entity header. In the figures, an element, which can be encrypted, is annotated with .

Our meta-model does not prescribe any cryptographic methods, like specific hash or encryption algorithms. Instead, when implementing the container format on the technical level, a catalogue with supported cryptographic algorithms needs to be embedded during application.

5 Conceptual Implementation

To map an automated analysis process of latent fingerprints to the generic process and build a process model, based on the meta-model, the following five steps are required:

Step 1 - Definition of data types for entity values. Entity values in the meta-model are treated as binary streams. To use the values in an actual process, they have to be associated with a data type, which is supported from the process. A global unique identifier is assigned to every data type to allow referencing from any number of containers. All data types have to be defined at first, to be referenced later.

Step 2 - Definition of data types for parameter sets. Equivalent to step 1, the data types for the parameter sets of the entity provenances must be defined.

Step 3 - Definition of the entity types, referencing data types for the values. Based on the data types, the entity types must be defined. Essentially an entity type is a list of property descriptions in the form (name, data type reference, use). The name is a string, unique in the context of the entity type. The data type reference points to a data type, defined in step 1. And the use can be "required" or "optional".

Step 4 - Definition of the data source interfaces, referencing an output entity type and optional a data type for a parameter set. To allow the process to start, the interfaces of all data sources, which are capable of producing entities, must be defined. A global unique identifier is assigned to every data source to be referenced from the entities in any number of containers. The output of the data source is specified by referencing an entity type, defined in step 3. Thus, the required and allowed entity values, produced by the source, are specified. If a data source is configurable by a number of parameters, its definition can reference a data type, defined in step 2, to store its configuration as provenance parameter set, along with the entity values.

Step 5 - Definition of the transformation algorithm interfaces, referencing input entity types, an output entity type and optional a data type for a parameter set. The interfaces of all transformation algorithms in the process must be defined. Therefore a global unique identifier is assigned to every transformation algorithm, accordingly to the data sources. The input of the algorithm needs to be specified in terms of a list, referencing entity types from step 3. The output type and optional a parameter set is defined in the same way as for the data sources in step 4.

Fig. 4 shows the overall method to implement an actual process model (left) and an actual container format (right) as well, which will be described later.

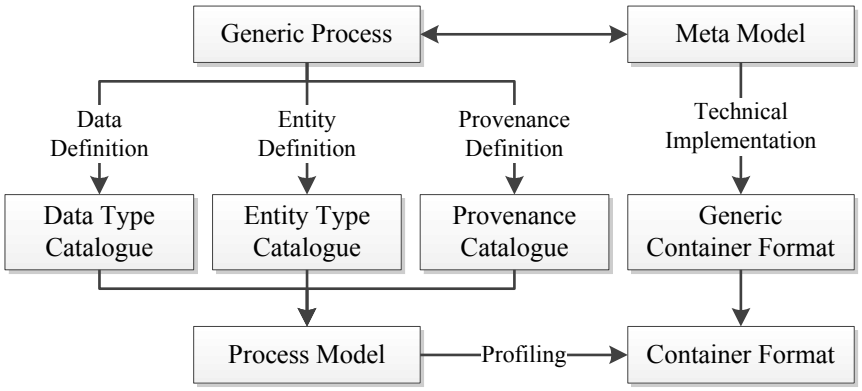

Fig. 4. Definition of an actual container format

The results are three catalogues:

Data types. This catalogue maps a global unique identifier to a data type description, which allows the interpretation of a binary data stream. The data can belong to an entity value or to an entity provenance parameter set.

Entity types. This catalogue maps a global unique identifier to a property list with references to data types, which allows the interpretation of the values in an entity.

Entity provenance interfaces. This catalogue maps a global unique identifier to an interface description of an entity provenance (sources and transformation algorithms), including input and output entity types and optional a data type for a parameter set. The relations between entity types and entity provenance interfaces are implementing the process model.

The following example illustrates an actual process model. The scenario is a fingerprint analysis process starting with a scanner, which delivers the raw image in a proprietary format. The scanner is the entity source and the proprietary raw data format is a value type, which is used by the entity type "RAW Data". There are two transformations taking "RAW Data" as input: The "Image Extraction" and the "Advanced Extraction". The algorithm for image extraction delivers an entity with the type "Bitmap"; the algorithm for advanced extraction reads the proprietary meta-data and delivers a "Physical Context" entity. In this simple example there is only one preprocessing algorithm, which takes a bitmap and delivers a bitmap. Another transformation is "Feature Extraction", which takes a "Bitmap" and a "Physical Context", and delivers a "Minutiae" entity, which encapsulates a minutiae feature vector. A possible analysis graph is shown in Fig. 5.

- Entity type catalogue: RAW Data, Bitmap, Physical Context, Minutiae
- Entity provenance catalogue
    - Scanner: input = (), output = "RAW Data"
    - Image Extraction: input = ("RAW Data"), output = "Bitmap"
    - Advanced Extraction: input = ("RAW Data"), output = "Physical Context"
    - Pre-Processing: input = ("Bitmap"), output = "Bitmap"
    - Feature Extraction: input = ("Bitmap", "Physical Context"), output = "Minutiae"

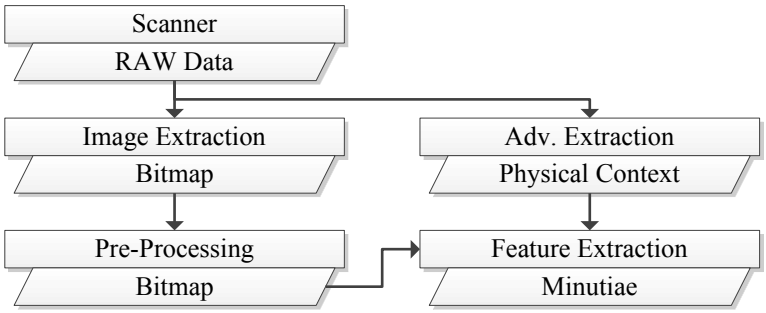

Fig. 5. Example of a multi-branched process graph

6 Technical Implementation

To build an actual container format for our concept, introduced in the previous section, a technical implementation of the meta-model is needed (see Fig. 4). In the following the basic information for a possible implementation, under use of current technologies, is given. However, this description cannot fulfill the task of a detailed technical specification.

The technical implementation stages in three levels. The first level is a storage mechanism, which can store files associated by a relative path. We suggest using the two storage mechanisms described in [10]: a directory in a file system and alternatively a ZipFile. The storage as a directory comes with the benefit of having very easy access to all elements in the container stored as individual files, but with a lack of compression while transport, e.g. as a HTTP download. The storage as a ZipFile comes with the benefit of compression and easier handling while transport, but with the requirement to use a programming library with ZipFile parsing capability while accessing the content of the container.

The second level is to define the way of structuring and storing elements of the meta-model in files and directories. To store entity values and parameter sets according to their associated data types, we propose to store them as binary streams in a file per value or parameter set, respectively. To store the other elements of the meta-model we propose to use XML files. We suggest one XML file for the container element as root, including the current edition, the history and the entity index, but excluding the entities and their child elements. Further, every entity including header, entity signature, parameter set reference and value references, is stored in its own XML file. The entity index, however, contains copies of the signatures from every entity. This is building a three-level-hierarchy: A XML file describing the container at the root, a XML file per entity and a binary file for every value or parameter set. To simplify the identification of value and parameter set files we propose to use a sub directory for every entity (see Fig. 6). The structure of the XML files for the container and the entities is defined by a XML schema [15, 16].

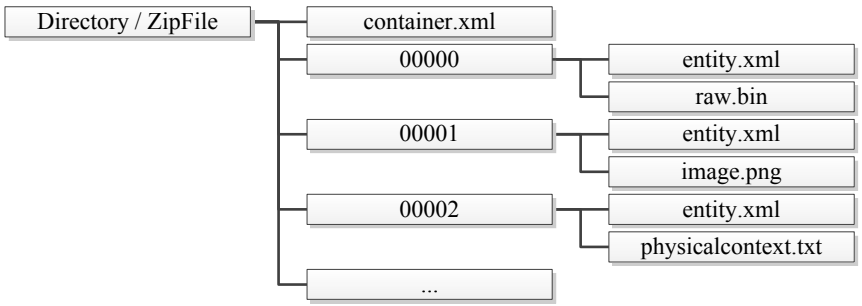

Fig. 6. Technical structure of a container

The third level of the implementation is to define the details. An important part is the design of a global unique identifier, e.g. to identify an edition. We propose to use GUID's after the scheme in RFC 4122 [17]. The signatures of the XML files, the

entity signature and the master signature, should be implemented by using the W3C recommendation for signatures in XML documents [18]. To build hashes from a XML file, the canonical form without comments [19] should be used. Another important part is to implement the cryptographic aspects of the format. We propose to define a catalogue with supported cryptographic algorithms for hashing as well as for encryption. That way, the technical implementation of the container format does not need to be changed, if a better cryptographic method is going to be used after the specification of the format is completed.

7 Conclusion and Future Work

This paper motivates to the requirement of a chain-of-custody-preserving processing framework for automated forensic analysis of fingerprints. It introduces a novel process model for the automated forensic analysis of latent fingerprints and proposes a framework to specify a container format for storing all necessary data of such a process to assure the chain of custody. The framework consists of a meta-model which describes the logical elements of a container format, a description how to instantiate an actual process model based on the meta-model and a draft of a way to implement an appropriate container format on the technical level.

The meta-model further assures the authenticity and integrity of every process step, including the data from the scanner over intermediate data to the resulting biometric features. With respect to an adaptive and modular analysis process, the meta-model supports a multi-branched process graph. To suit the need for confidentiality the stored data can be encrypted. Further the draft of the technical implementation for a container format suggests the use of current technologies, like XML and ZipFiles. Following this draft, the data stored in a container can be extracted easily with a wide variety of programming languages or scientific software packages like GNU Octave and is thereby easy to use in research and application.

In future work, a detailed specification for the technical implementation needs to be devised. The implementation of the ZipFile version of the container format can benefit from the work in AFF4 [10], especially when large data values are used. Furthermore, two catalogues with standard data types and standard entity types for biometric applications are desirable. These catalogues can make benefit from the work in the CBEFF format [11], and XCBF format resp. [12]. At last a reference implementation for the container format is planned within the scope of an actual research project.

Acknowledgments

The authors want to acknowledge the support of all research partners in the project, especially Mario Hildebrand and Ronny Merkel of Otto von Guericke University.
This work is supported by the German Federal Ministry of Education and Research (BMBF), project “Digitale Fingerspuren (Digi-Dak)” under grant number 13N10816. The content of this document is under the sole responsibility of the authors.

References

1 Digitale Fingerspuren, http://omen.cs.uni-magdeburg.de/digi-dak/ (2010)
2 Bishop M.: Introduction to Computer Security. Addison-Wesley Longman, Amsterdam. ISBN 0321247442. (2004)
3 Garfinkel S.L.: Providing cryptographic security and evidentiary chain-of-custody with the advanced forensic format, library, and tools. In: IJDCF vol. 1.1, pp. 1--28. (2009)
4 Raul A.C., Dwyer J. Z.:"Regulatory Daubert": A Proposal to Enhance Judicial Review of Agency Science by Incorporating Daubert Principles into Administrative Law. In: Law and Contemporary Problems, vol. 66.4, pp. 7--44 (http://www.law.duke.edu/shell/cite.pl ?66+Law+&+Contemp.+Probs.+7+%28Autumn+2003%29), (2003)
5 Octave, http://www.gnu.org/software/octave/ (2010)
6 Zhang D.D.: Automated biometrics: Technologies and systems. pp. 8--10, Kluwer, Boston (2000)
7 Vielhauer C.: Biometric User Authentication for IT Security: From Fundamentals to Handwriting. Springer, New York (2006)
8 Garfinkel S.L.: Afflib, http://afflib.org/ (2010)
9 Keightley R.: EnCase version 3.0 manual revision 3.18, http://www.guidancesoftware .com/ (2003)
10 Kohen M., Garfinkel S.L., Schatz B.: Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow. In: Digital Investigation vol. 6.1, pp. 57--68. Elsevier (2009)
11 Podio F.L., et.al.: Common Biometric Exchange Formats Framework, NIST IR 6529, http://csrc.nist.gov/publications/nistir/NISTIR6529A.pdf. NIST, Gaithersburg (2004)
12 Larmouth J.: XML Common Biometric Format, http://www.oasis-open.org/specs/. OASIS (2003)
13 ISO/IEC 19794:2007: Information Technology: Biometric Data Interchange Formats, International Organization for Standardization, Geneva, Switzerland.
14 Merkle R.C.: A certified digital signature. In: Advances in Cryptology (CRYPTO89) Santa Barbara, Lecture notes in computer science, vol. 435, pp. 234--246, Springer, New York (1989)
15 W3C: XML Schema Part 1: Structures Second Edition, http://www.w3.org/TR/xmlschema-1/ (2004)
16 W3C: XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/xmlschema-2/ (2004)
17 Leach P., et.al.: A universally unique identifier (UUID) URN namespace. RFC 4122. http://www.ietf.org/rfc/rfc4122.txt (2005)
18 W3C: XML Signature Syntax and Processing Second Edition, http://www.w3.org/TR/xmldsig-core/ (2008)
19 W3C: Canonical XML Version 1.0, http://www.w3.org/TR/2001/REC-xml-c14n-20010315 (2001)