Database-Centric Chain-of-Custody in Biometric Forensic Systems

Martin Schäler, Sandro Schulze, and Stefan Kiltz

School of Computer Science, University of Magdeburg, Germany
{schaeler,sanschul,kiltz}@iti.cs.uni-magdeburg.de

Abstract. Biometric systems gain more and more attention in everyday life regarding authentication and surveillance of persons. This includes, amongst others, the login on a notebook based on fingerprint verification, controlling of airports or train stations, and the biometric identity card. Although these systems have several advantages in comparison to traditional approaches, they exhibit high risks regarding confidentiality and data protection issues. For instance, tampering biometric data or general misuse could have devastating consequences for the owner of the respective data. Furthermore, the digital nature of biometric data raises specific requirements for the usage of the data for crime detection or at court to convict a criminal. Here, the *chain-of-custody* has to be proven without any doubt. In this paper, we present a database-centric approach for ensuring the chain-of-custody in a forensic digital fingerprint system.

1 Introduction

When using physical evidence in law enforcement proceedings, a so called *chain-of-custody* has to be maintained in order for that evidence to be admissible in court. According to [19], at a crime scene evidence must be preserved for court use. Also, a documentation suitable for federal, state and local courts must be developed. It is the duty of the first responder of law enforcement to ensure that all evidence is protected and documented. Along with this, the *chain-of-custody* starts, which describes the route that evidence takes from its initial possession until its final disposition. In that process, a proper documentation process is of highest importance. It has to be proven without doubt that the evidence is *authentic* and *holds integrity*, that is, the evidence is original and has not been tampered with.

This applies also to digital evidence as used in IT-Forensic. The security aspects of *authenticity* and *integrity* have to be assured. Authenticity, according to [10], can be divided into two aspects: First, data origin authenticity is the proof of the data's origin, genuineness, originality, truth and realness. Data authenticity requirements can also be defined as prevention, detection, and recovery requirements. Second, entity authenticity is the proof that an entity, like a person or other agent, has been correctly identified as originator, sender or receiver. Hence, it can be ensured that an entity is the one it claims to be. Both, data and entity authenticity, are relevant in law enforcement proceedings. Beyond that, the security aspect of integrity (see also [3]) refers to the integrity of resources. It describes whether a resource (e.g., information) is altered or manipulated. Hence, integrity is the quality or condition of being whole and unaltered, and it refers

to the consistency, accuracy, and correctness of the resource. Furthermore, the security aspect of *confidentiality* plays an important role in law enforcement proceedings. In detail, confidentiality refers to information that needs to be treated secret from unauthorized entities [3]. A special aspect of confidentiality is privacy, where person related data needs to be protected.

As described in [13], cryptographic mechanisms can be used to ensure the chain-of-custody and therefore, inherently, authenticity and integrity. This however, does not only apply to forensically relevant data gathered by investigating IT-based incidents. Also the traditional criminal investigation process today can be supported, e.g. by the usage of contact-less forensic fingerprint scanners as suggested in [17]. By using techniques also applied in biometric applications that allow authorization and access, digitally represented fingerprint data from a physical origin can play a very important role in law enforcement and court proceedings. For such biometric data, also a digital chain-of-custody needs to be maintained, following the same strict rules as with physical evidence. Additionally, fingerprint data are personal data, for which stringent laws exist in some countries, e.g. in Germany the federal data protection act (Bundesdatenschutzgesetz [11]). Especially for fingerprint data, the security aspect of confidentiality is very important, since such data must not be made available to unauthorized users.

In [19] it is suggested that cryptographic means are applied to data that is represented in a file system. In the approach presented in the following we investigate the appropriateness of the mechanisms provided by a database system and what extra mechanisms have to be applied.

Our contribution in this paper is as follows. Initially, we propose a database-centric chain-of-custody for relational [7] and object-relational [20] database systems. This includes a formalization of a *reliable data provenance concept* and additional requirements for an implementation of the provenance concept to ensure authenticity and integrity of the data. Furthermore, we show how the concept is integrated tightly into a fingerprint verification database to make it's circumvention as hard as possible. Finally, we exemplary show how our approach can be used to prevent malicious modification and to detect the circumvention of the chain-of-custody.

2 Problem Statement

In a current research project, we explore pattern recognition techniques for fingerprints that are captured by contact-less, optical surface sensors. To this ends, we develop a fingerprint verification database (*FiVe DB*) to support this recognition process. Although FiVe DB supports other research tasks as well, such as evaluating sensor techniques for capturing latent fingerprint, the main focus is on verifying whether a certain digital data item (captured by a sensor) contains a fingerprint or not. Since this verification process involves different transformations (e.g., for quality enhancement) of the original sensor data, the authenticity, integrity and confidentiality of the data must be guaranteed throughout the process. Otherwise, the usage of the data (e.g., as a proof at court) may cause legal problems. We conclude that our database must apply to the chain-of-custody to ensure the integrity and authenticity of the fingerprint data.

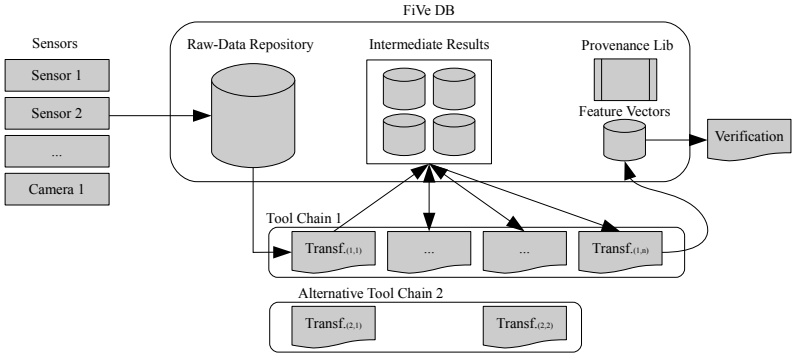

Fig. 1. Holistic Infrastructure

FiVe DB in the center of the global architecture. To clarify, why the database is responsible for the chain-of-custody, we introduce a simplification of our architecture in Figure 1. The potential fingerprints are captured by some kind of sensor. That might be a simple digital camera or some other kind of scanner. We call this first digital image of fingerprint, which is stored in FiVe DB, a *raw data item*. Generally, the raw data format of two different sensors can differ, e.g., one sensor could provide additional topographic data. Furthermore, several transformations can be applied to a raw data item (or an intermediate result of a previous transformation) and thus create a new intermediate result. Each intermediate result is stored in FiVe DB persistently. Afterwards, it is possible to perform another transformation on the specified data. As a result, transformations are always performed on data items, stored in the database. Hence, we call this architecture *database-centric*. The final transformation creates a *feature vector* that allows some kind of verification to decide whether there is a fingerprint or not. Above all, a feature vector remains an image with some extracted data (e.g. material information, is there a fingerprint, is there an overlapped fingerprint etc.). Furthermore, the database contains functionality to compare different transformation chains and sensor techniques. Every piece of data in the DB is created *once* and not modified again. Here, it is worth to mention that we *do not* provide any solution of automatically identifying a suspect by its latent fingerprint.

In Figure 2, we show how the data items and its corresponding transformations are stored internally in form of tree structures. The root of a certain tree is a raw data item. The leaf is either a feature vector or some intermediate result. We call every path from the root to the leaf a *fingerprint data set (FpD)*. This serves as our central data unit for which we have to prove the chain-of-custody. The single chain links are the single data items, which are created by a sensor or quality enhancement transformations.

Relationship between chain-of-custody and data provenance. The term chain-of-custody from a biometrics point of view is highly related (but not equivalent) to data provenance in the database domain [22]. In databases, data provenance provides information about the origin of data and the transformation performed on these data

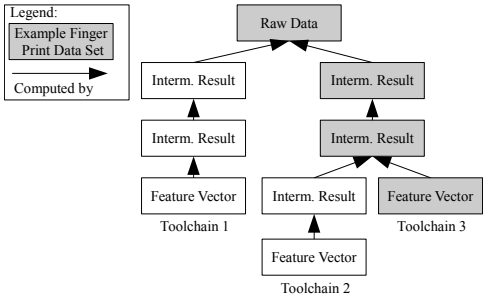

Fig. 2. Fingerprint Data Set (FpD)

items [5,6]. Although this is exactly the information we are interested in for ensuring the chain-of-custody, the mechanisms, which are usually used for data provenance, do not take *reliability* into account. This means that we cannot guarantee authenticity and integrity. For instance, to provide information on the original source of data, *foreign keys*, that is, referencing data by its unique identifier, can be used as a pointer to the original raw data item. Furthermore, history tables can be used to log the transformation chain. Unfortunately, foreign keys can easily be modified so that it points to the wrong original sensor image. In the same way, history and log tables can be tampered to obfuscate unauthorized or unproper changes [23]. As a result, authenticity and integrity of the data items are violated. In a real forensic scenario this may lead to devastating consequences.

Consequently, we need a *reliable data provenance approach* for relational databases that creates a chain-of-custody and thus guarantees *integrity*, *authenticity* and *confidentiality*. We must be able to prevent and detect tampering of the data that are described in the next subsection. Since no database mechanisms exist to ensure authenticity and integrity of the provenance information by our means, we have to develop additional countermeasures and integrate them into the database.

2.1 Attacker model

To find the appropriate countermeasures, as a step of a risk analysis (see [21]) we must analyze *which* attacks are possible on our system and *how* they affect the authenticity, integrity and confidentiality of the data. According to Figure 1, we see the following threats (see [16]) of spoofing, modifying, reading or deleting data in the holistic infrastructure that we need to prevent and detect whenever the prevention was circumvented.

1. Faking a raw data item by a sensor,
2. Tampering the data send from a sensor to FiVe DB,
3. Any modification of some piece of data in FiVe DB,
4. Tampering the data send from FiVe DB to a transformation,
5. Faking an intermediate result or feature vector by a transformation,
6. Tampering a result send from a transformation to FiVe DB.

The attacks one and five try to insert unauthentic, but possibly data that hold integrity into FiVe DB. By contrast, in every other attack the integrity of the data is affected, but

they may be related to an authentic fingerprint image. In the following, we discuss to which extent the mentioned threats can be addressed by the suggested concept and which possible threads cannot be addressed by our concept.

3 Formalization of the Provenance Concept

In this section we present a formalization of the *provenance concept* that lays the foundation for the chain-of-custody of our architecture. Furthermore, such a formalization is independent of a concrete implementation (i.e., how the concept is realized in a certain system). We also define further requirements that an implementation of the provenance mechanism has to fulfill so that the chain-of-custody can be proven without doubt.

3.1 Definition

First, we need to specify how a fingerprint is represented in the system to know which data items are subject to provenance information. Note that the formalization of the provenance concept can be reused in any kind of biometric application and is not limited to our system and purposes.

Definition 1. *A Fingerprint Data Set (FpD) consists of several binary data items (D). The raw data (D_{raw}) is the original data from the sensor. The sequence of intermediate results S(D_{ir}) contains the data after each transformation and the feature vector (D_{fv}) is the final result of a transformation chain.*

$$FpD = \{D_{raw}, S(D_{ir_1}, \ldots, D_{ir_n}), D_{fv}\} \tag{1}$$

Each D contains some kind of redundant structured data that allows to use them with common SQL, the query language commonly used in relational database systems. Generally, our formalization is independent of the data format, the result of a transformation or the representation of the feature vector, because we treat them as some kind of binary data. As a result, our concept is flexible regarding the structure and semantic of the data, but we cannot semantically check whether the binary data is correct on the data itself. For instance, imagine a transformation that applies a Garbor Filter to an intermediate result. In this case, FiVe DB does not check whether the result of this transformation is an image again. Particularly, FiVe DB tests whether the (trusted) transformation delivers readable provenance information that indicates *how* the transformation computed the result. Hence, we have to rely on the associated provenance information to ensure the chain-of-custody. The granularity of the transformations can be chosen as needed by the underlying system.

As an refinement of our initial definition, we define how a feature vector (the final result of a transformation chain) is calculated from the raw data by a sequence of transformations. Note that a feature vector remains an image with some extracted data as previously stated in Section 2.

Definition 2. *A Transformation is an operation which creates a new binary data item from a different one. The feature vector is calculated by a finite sequence of transformations from one raw data item. The results (if not equivalent to D_{fv}) are called*

intermediate results (D_{ir}) which are stored in the database.

$$t\colon D \rightarrow D \qquad\qquad D_{fv} = t_n(\ldots(t_0(D_{raw}))) \tag{2}$$

Since our overall formalization is independent of any realization, the definition below abstracts from the semantics of a concrete transformation. Hence, we can deal with different transformations, used for different purposes such as tool chain or sensor evaluation, in the same way. However, for a concrete realization, the transformation must be certified so that we can trust its implementation, because, as previously mentioned, we cannot semantically check whether the result of a transformation is correct.

As a result of the previous definitions, we must have knowledge of the corresponding raw data item for each piece of data[1] as well as the transformation chain that lead to this item, the previous intermediate result and the meta data (e.g., the source such as a scanner, the creation date etc.) to ensure authenticity and integrity of one data item. We call this information a *ProveSet* and it is formalized as follows.

Definition 3. *A ProveSet for some piece of data D_k is defined as a tuple of a link to the raw data $L(D_{raw})$ it stems from, a sequence of transformation that created D_k from D_{raw}, a link to the previous intermediate result D_{k-1} and the meta data M that is provided by the sensor.*

$$ProveSet(D_k) = \{L(D_{raw}), S(t_0, \ldots, t_{k-1}), L(D_{k-1}), M\} \qquad D_k = t_{k-1}(D_{k-1}) \tag{3}$$

In the ProveSet of a raw data item, the sequence of transformations and the predecessor D_{k-1} is empty. In the ProveSet of a feature vector the complete transformation sequence is available.

Finally, we need to know the ProveSet for each data item of a fingerprint data set (FpD) to verify that the whole storage and transformation process performed well and to ensure that the FpD can be used in court without any doubt regarding authenticity and integrity. Therefore, we extend our ProveSet definition and define the *Complete ProveSet (CProveSet)* as follows.

Definition 4. *A Complete ProvSet (CProveSet) for an FpD exists, if for each data item D_k of the FpD the corresponding ProveSet exists.*

$$CProveSet(FpD) \leftrightarrow \forall D_k \in FpD\ \exists\ ProvSet(D_k) \tag{4}$$

A CProveSet is *correct* when there are no contradictions within the ProvSets of the single data items. With this formalization, we can track the whole history of an FpD, but to ensure authenticity and integrity we need to define additional requirements for the provenance mechanism.

3.2 Additional requirements for the provenance mechanism

As mentioned before, we have to rely on the provenance information, so the mechanism has to prevent unauthorized change of the data as well as deleting or modifying the

[1] This piece is either a raw data item, an intermediate result or a feature vector.

provenance information. For this reason, we define the following requirements, that must be fulfilled by the implementation of the provenance concept. It is worth to mention that the *residential risk* of circumventing this chain-of-custody is highly dependent on its implementation and the used database system. Consequently, we eventually need an additional intrusion detection system or we have to use IT-Forensics.

Change detection. Every change in a data item must be detectable. This means that an unwanted or malicious modification (including deletion) of a data item (D_k) of an FpD must be detectable.

Tight coupling. The binary data of a data item (D_k) and the corresponding $ProveSet(D_k)$ must be tightly coupled, so that it is *practically impossible* to delete or modify the ProveSet of D_k in an *unauthorized* manner.

Applicability. We need some functionality that allows to check the ProveSet of any data item by the database itself.

Performance. The overhead of verifying the ProveSet and the additional needed disk space to store the ProveSet shall be reduced to a minimum.

Modification. A transformation t_k has to have some possibility to extend the ProveSet(D_k) of its input data D_k to create the ProveSet(D_{k+1}) of its output data D_{k+1} as follows:

$$\begin{aligned} ProveSet(D_k) &= \{L(D_{raw}), S(t_0, \ldots, t_{k-1}), L(D_{k-1}), M\} \\ ProveSet(D_{k+1}) &= \{L(D_{raw}), S(t_0, \ldots, t_{k-1}, t_k), L(D_k), M\} \end{aligned} \tag{5}$$

4 Our Solution

In this section we present a solution that integrates the provenance concept described in Section 3.1 into FiVe DB and fulfills the additional requirements from Section 3.2.

4.1 The provenance mechanism - Verifying structured data with redundant unstructured data

Subsequently, we show how a data item D_k and its ProveSet(D_k) stick together, so that FiVe DB can check them. Additionally, we explain the realization of the additional requirements.

Connecting the provenance information in the ProveSet tightly to the data. As described in Section 3.2, we need a mechanism that connects the ProveSet tightly to the data. In our solution, the ProveSet is embedded into the binary data of D_k. That means that whenever a transformation requests some D_k, the corresponding ProveSet is delivered as well. This is realized by the data format illustrated in Figure 3.

Every data item D_k has a part that contains redundant structured data that makes them *applicable* with common SQL and a part, containing unstructured data including the ProveSet. Redundant in this context means that structured data, such as the foreign key of the original raw data item, is also embedded into the binary unstructured data. Due to this separation, the performance of the system is still reasonable because the ProveSet is not extracted form the binary data whenever queries are performed on the structured

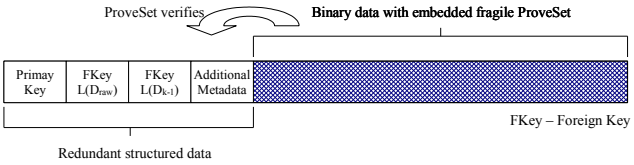

Fig. 3. Format of a data item

part of the data item. Additionally, we prevent from modification of the redundant data by mechanisms of the database system. It is worth to remind that the data items are inserted in FiVe DB and not modified again. When we need to verify the structured information we can do that by calling a stored procedure *check(primary_key)* from the provenance library (see Figure 1) that extracts the *fragile* ProveSet(D_k) from the binary (unstructured) data. If the *check()* function fails (i.e., it is not possible to extract the ProveSet), this indicates that the binary data has been changed unauthorized. If there are contradictions between the extracted ProveSet and the redundant structured data, the structured data has been modified unwarranted. So we can *detect* inappropriate *modification* of the data and determine that the data no longer holds integrity. We integrate the integrity checks into the system's processes that are explained in Section 4.2. Furthermore, we create internal database jobs that execute the checks continuously on the whole FiVe DB.

To realize the embedding of the fragile ProveSet into the binary data, we plan to use an invertible watermarking technique [9]. The watermark is embedded into the binary data of D_k. Furthermore, it is possible to embed the ProveSet into the watermark. Whenever an attacker modifies the data, the fragile watermark, including the ProveSet, cannot be extracted by FiVe DB. Hence, we can detect malicious data modification.

4.2 Integration of the Provenance Concept

One basic concept in designing FiVe DB is to integrate the provenance concept tightly into the system behavior to make the circumvention of this concept as hard as possible. Another important issue is to detect every inappropriate modification of an FpD or its ProveSet. As illustrated in Figure 1, there are three operations that may add data to FiVe DB: insert a new raw data item, create an intermediate result and calculate a feature vector. For each of these operations, we specify a process that ensures that the operation is well defined. We prevent every other (inappropriate) modification of the data by mechanisms of the database system itself. For example, we use a fine grained role system to specify who has access to what data. Access control is a native part of each database system following the SQL Standard [1]. Furthermore access control is one of FiVe DB's mechanisms to ensure confidentiality that is recommended for a chain-of-custody. In future we will have to define more processes for maintenance purposes where some attributes of a data item may be changed.

As an example, we will explain the process of creating a new intermediate result, because it is the most complex one and show how it fulfills the formalization presented in 3.1. The other two processes are designed in the same manner.

Example process - Inserting a new intermediate result. Creating a new intermediate result consists of three steps. First, a transformation requests an intermediate result or raw data item from FiVe DB. Second, the transformation computes an new intermediate result. Finally, this result with the appropriate provenance information is inserted into FiVe DB. We illustrate the process in Figure 4 in detail and describe the particular steps in the following. Thereby, we will explain how we guarantee integrity and authenticity.

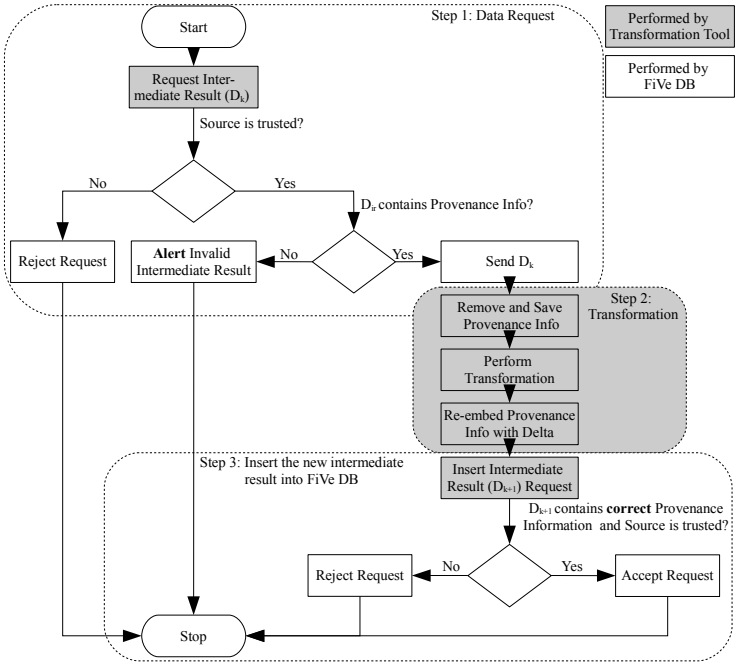

Fig. 4. Process example: Create a new intermediate result

Data request. A transformation initially triggers the process requesting the input data (D_k) from FiVe DB by calling a *stored procedure*. There is no direct way of accessing the (binary) data[2] via common SQL. This allows FiVe DB to check whether the requested data has the provenance information and to collect additional meta data.

In the case that FiVe DB trusts the transformation tool (*confidentiality*), FiVe DB checks whether the requested data D_k has a *ProveSet(D_k)*. Afterwards, the DB determines the corresponding FpD and checks *if CProveSet(FpD) is correct* using the provenance library (see Figure 1 and attack three in Section 2.1). Hence, we can guarantee that any previous transformation process from the raw data to D_k has performed well: the data holds *integrity*. This also means that FiVe DB can trace back the intermediate result to a raw data item that was inserted by a trusted sensor, so the data is *authentic*. In

[2] The tools have read-only access to the meta data to select the right input.

this way we assure that FiVe DB only sends intermediate results to a transformation tool that hold *integrity and authenticity*.

Transformation. In step two, the transformation tool first checks the existence of ProveSet(D_k). If it does not exist the data has been modified during the transportation from FiVe DB to the tool (see attack four in Section 2.1). When ProveSet(D_k) exists, it has to be removed from D_k to perform the transformation. The transformation tool now creates D_{k+1} as t(D_k) and embeds ProveSet(D_{k+1}) into D_{k+1}, which allows FiVe DB to check whether this intermediate result holds *integrity* (detect attack six).

Insert new intermediate result. Finally, the transformation tool calls a different stored procedure to insert the new intermediate result D_{k+1}. Before accepting the request FiVe DB checks whether the tool is trusted, so we can assume that D_{k+1} is computed as t(D_k) and Five DB knows that D_k is authentic (prevent attack five from Section 2.1). Furthermore FiVe DB tests whether ProveSet(D_{k+1}) exists and if the CProveSet(FpD) including D_{k+1} is correct, to verify the integrity of D_{k+1}. Thus, we can detected data in this process that does not hold *integrity or authenticity*.

Rejecting request and alerts. There are several routines in the processes that detect an incorrect CProveSet or react when an untrusted transformation or sensor tries to communicate with FiVe DB.

5 Related Work

When using database systems, the security aspects of authenticity and integrity as well as confidentiality must be met (see Section 1). Especially to ensure confidentiality, it must be assured that data no longer needed is securely deleted and that the data when in use is not accessible to unauthorized users. To detect and investigate security breaches, IT-Forensics has to be applied to the mass storage, the main memory and the network data stream of a computer system. For that, the operating system, the file system, the database system and any security software as explicit means of intrusion detection as well as any scaling of methods for evidence gathering and software for data processing and evaluation will have to be investigated for the presence of forensically relevant data. In [15], Kiltz et al. show this process, using a holistic model of the forensic process that differentiates into phases, classes of forensic methods and forensically relevant data types, how main memory content can be acquired, investigated and analyzed. In [12] Fowler shows, how a database system can be forensically investigated and analyzed at the example of the Microsoft SQL Server. Here it is stated that part of potential digital evidence can be found in the main memory and in the mass storage section of a computer system but to make sense of some of the data, methods of the IT-Application (i.e., the database system) have to be used.

Ensuring the chain-of-custody in databases for forensic systems in databases is a quite new research topic. The general data provenance mechanism in databases often deal with uncertain and structured data and are not reliable [2,5,8,23]. A reliable provenance mechanism based on watermarks to guarantee the chain-of-custody for digital images,

which may be used in court, is explained in [4]. This approach does not use databases and does not include quality enhancement of the original image. In [14] Hasan et al. present a secure provenance mechanism in documents, which can be interpreted as chain-of-custody. The authors concentrate on detecting unauthorized rewrites of documents and it's chain-of-custody and leave the question of efficiently tracking unauthorized reading attempts open. By contrast, we can identify unauthorized reading attempts, because of our database-centric architecture and the usage of stored procedures, which allow fine grained logging. Additionally, we can use native mechanisms of the database such as role-based access to ensure *confidentiality*. Furthermore, in our concept the provenance information (ProveSet) is tightly linked to the data, so whenever the data is send the ProveSet is delivered as well.

6 Conclusion and Future Perspectives

In this paper, we presented a database-centric chain-of-custody for databases in forensic biometric systems. First, we explained the importance of ensuring *integrity and authenticity* in forensic scenarios. We introduced FiVe DB as an example and we described an attacker model based on the architecture and the data stored in FiVe DB. To ensure the chain-of-custody, we explained that common mechanisms to track the history of a data item used in databases (data provenance) are generally *not reliable*, because they can easily be modified. Consequently, we developed a formalization of a provenance concept that provides the needed information. Additionally, we defined supplemental requirements that an implementation of this concept has to fulfill to be *reliable*. Furthermore, we explained a mechanism that allows to install our reliable provenance concept. This mechanism extends the general approach by storing redundant fragile data in the binary unstructured image of each data item. Finally, we showed how we integrate the provenance concept tightly into FiVe DB to ensure integrity and authenticity. Hence, the circumvention of the chain-of-custody is as hard as possible and we can detect malicious modification of the data (with a certain *residential risk*).

In future, we plan to evaluate different implementation of our provenance concept on different database systems according to the requirements defined in Section 3.2. Additionally, we have to extend the concept, by defining processes to modify the data for maintenance purposes, which alters the attacker model (see Section 2.1). Furthermore, we want to examine data fusion algorithms to support efficient query computation.

7 Acknowlegements

The work in this paper has been funded in part by the German Federal Ministry of Education and Science (BMBF) through the Research Program under Contract No. FKZ: 13N10817 and FKZ: 13N10818. We would also thank Prof. Dr.-Ing. Jana Dittmann and Prof. Dr. Gunter Saake for their support for this paper.

References

1. ANSI/ISO/IEC 9075:1999. *International Standard - Database Language SQL*, 1999.

2. O. Benjelloun, A. Sarma, A. Halevy, and J. Widom. Uldbs: databases with uncertainty and lineage. In *Proc. Int. Conf. on Very Large Data Bases*, pages 953–964. VLDB, 2006.
3. M. Bishop. *Computer Security - Art and Science*. Addison-Wesley, 2003.
4. P. Blythe and J. Fridrich. Secure digital camera. In *Proc. of Digital Forensic Research Workshop*, pages 17–19, 2004.
5. P. Buneman, S. Khanna, and W.-C. Tan. Why and where: A characterization of data provenance. In *Proc. Int. Conf. on Database Theory*, pages 316–330. Springer, 2001.
6. J. Cheney, L. Chiticariu, and W.-C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.
7. E. Codd. A relational model of data for large shared data banks. *Comm. of the ACM*, 13(6):377–387, 1970.
8. Y. Cui, J. Widom, and J. Wiener. Tracing the lineage of view data in a warehousing environment. *ACM Trans. Database Syst.*, 25:179–227, 2000.
9. J. Dittmann, S. Katzenbeisser, C. Schallhart, and H. Veith. Provably secure authentication of digital media through invertible watermarks. *Cryptology ePrint Archive, Report 293*, 2004.
10. J. Dittmann, P. Wohlmacher, and K. Nahrstedt. Using cryptographic and watermarking algorithms. *IEEE MultiMedia*, 8:54–65, 2001.
11. The Federal Commisioner for Data Protection and Freedom of Information. Federal data protection act (bdsg) in the version promulgated on 14 january 2003 (federal law gazette i, p. 66), last amended by article 1 of the act of 14 august 2009 (federal law gazette i, p. 2814), in force from 1 september 2009.
12. K. Fowler. *SQL Server Forensic Analysis*. Addison-Wesley, 2008.
13. S. Garfinkel. Providing cryptographic security and evidentiary chain-of-custody with the advanced forensic format, library, and tools. *Digital Crime and Forensics*, 1(1):1–28, 2009.
14. R. Hasan, R. Sion, and M. Winslett. The case of the fake picasso: preventing history forgery with secure provenance. In *Proc. Int. Conf. on File and Storage Technologies*, pages 1–14. USENIX Association, 2009.
15. S. Kiltz, T. Hoppe, J. Dittmann, and C. Vielhauer. Video surveillance: A new forensic model for the forensically sound retrival of picture content off a memory dump. In *Proc. of Informatik 2009 - Digitale Multimedia-Forensik*, volume 154 of *LNI*, pages 1619–1633, 2009.
16. S. Kiltz, A. Lang, and J. Dittmann. Taxonomy for computer security incidents. In *Cyber Warfare and Cyber Terrorism*, 2007.
17. M. Leich and M. Ulrich. Forensic fingerprint detection: Challenges of benchmarking new contact-less fingerprint scanners – a first proposal. In *Proc. Workshop on Pattern Recognition for IT Security*, Darmstadt, 2010. TU-Darmstadt.
18. M. Meints, H. Biermann, M. Bromba, C. Busch, G. Hornung, and G. Quiring-Kock. Biometric systems and data protection legislation in germany. In *Proc. Int. Conf. on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1088–1093. IEEE, 2008.
19. R. Newman. *Computer forensics: evidence, collection, and management*. Auerbach, 2007.
20. M. Stonebraker and D. Moore. *Object Relational DBMSs*. Morgan Kaufmann, 1996.
21. G. Stoneburner, A. Goguen, and A. Feringa. *Risk management guide for information technology systems [electronic resource] : recommendations of the National Institute of Standards and Technology*. U.S. Dept. of Commerce, National Institute of Standards and Technology.
22. W.-C. Tan. Provenance in databases: Past, current, and future. *IEEE Data Engineering Bulletin*, 32(4):3–12, 2007.
23. J. Zhang, A. Chapman, and K. LeFevre. Do you know where your data's been? - Tamper-evident database provenance. Technical Report CSE-TR-548-08, Univ. of Michigan, 2009.