UNIVERSITY OF TARTU
Institute of Computer Science
Cyber Security Curriculum

Deivis Treier

Research and Proof of Concept of Selected ISKE Highest Level Integrity Requirements

Masters's Thesis (30 ECTS)

Supervisor: Raimundas Matulevičius, PhD

Tartu 2017

Research and Proof of Concept of Selected ISKE Highest Level Integrity Requirements

Abstract: Information security becomes more and more important in today's society, where more processes and operations will be digitised and data moves from paper to bits and bytes and receives digital form. In Estonia state and public institutions are collecting and processing information for providing high level services, fulfilling state needs on constitutional tasks or international contracts. The public sector in Estonia must apply information security standard IT Baseline Security System ISKE requirements in three factors: availability, integrity and confidentiality of processed data.

This thesis takes integrity domain under detail research to meet ISKE requirements and security objectives demanded for data with highest integrity needs. By analysing the integrity domain of ISKE and providing versatile proof of concept about solution for implementing security controls, it is possible to increase awareness of software developers and ISKE implementation participants to achieve better information security.

Keywords: ISKE, information integrity, applied cryptography, data tampering detection, NoSQL, DBMS, Information Security Risk Management, ISSRM, software development, requirements validation

CERCS: P170

Valitud ISKE kõrgeima terviklikkuse nõuete teostuse uurimus ja kontseptsiooni tõestamise projekt

Lühikokkuvõte: Informatsiooni turvalisus on saamas üha olulisemaks tänapäeva ühiskonnas, kus üha rohkem protsesse ja tegevusi digitaliseeritakse ja andmed liiguvad paberilt bittideks ja baitideks digitaalsele kujule. Eesti riigi- ja avalikud asutused koguvad ja töötlevad informatsiooni, et tagada kõrgetasemelisi teenuseid, täita põhiseaduse kohustusi või rahvusvahelisi lepinguid. Avalik sektor Eestis peab täitma andmete käitlemisel informatsiooni turvalisuse standardi infosüsteemide turvameetmete süsteem ISKE nõudjeid kolmes teguris: käideldavus, terviklus ja konfidentsiaalsus.

Magistritöö võtab tervikluse valdkonna detailsema uurimise alla, et saavutada ISKE meetmete ja turvaeesmärkide täitmine, mis on nõutud kõrgeima terviklusega andmetele. Analüüsides ISKE tervikluse valdkonda ja luues mitmekülgse kontseptsiooni teostuse tõestamise projekti turvanõuete realiseerimise meetmetele on võimalik suurendada arendajate ja ISKE rakendamise partnerite teadlikkust saavutamaks parem informatsiooni turvalisus.

Võtmesõnad: ISKE, informatsiooni terviklus, rakenduslik krüptograafia, andmete omavolilise muutmise avastamine, NoSQL, DMBS, informatsiooni turvalisuse riski haldamine, ISSRM, tarkvara arendus, nõuete valideerimine

CERCS: P170

Acknowledgments

This thesis project was the most dedicated research I have carried trough. The focus of this research paper was one of the reasons I started in the Cyber Security curriculum.

I would like to thank the following people and institutions for helping me during my research project. I would like to thank my supervisor Raimundas Matulevičius, PhD (University of Tartu) for giving me comprehensive guidance and motivating me in this long journey.

Special thanks to my ex-colleagues in the IT and Development Centre, Ministry of the Interior for meaningful discussions in the beginning of the research project.

For testing and validating the solution I used a hardware cryptographic tool and a test certificate for qualified digital signature (e-seal) provided by SK ID Solutions AS. I thank Tiia Tampere (SK ID Solutions AS) and Eneli Kirme (SK ID Solutions AS) for providing me comprehensive support in the topic of hardware secure module topic.

For language review I would like to thank Karolin Lillemäe.

Finally I would like to thank you - the reader of this thesis.

Contents

1 Introduction

In this section we provide an introduction to the thesis and the motivation behind this research work. It gives an overview of Estonia's IT Baseline Security System ISKE, the scope of the research work and finally we give the outline of this thesis.

1.1 Background

Estonia's IT Baseline Security System hereinafter ISKE [1] is an information security standard developed for the Estonian public sector. It can also be understood as a framework for the security of information assets. For state and local government organisations who process public data, it is obligatory to follow rules specified by ISKE [2]. ISKE's application process has been drawn up on the basis of long-term practice of security analysis and management of typical information assets [2][3].

ISKE is based on the German information security manual BSI - IT Baseline Protection Manual (*IT-Grundschutz*), what was adapted to Estonian circumstances [2].

ISKE is instructed by its Reference Guide [3] what contains a guide and tools to determine the security level of information assets according to safety class. Depending on the degree of security requested, security measures, requirements and controls are specified in the Requirements Catalogue [4]. Risks are determined in the Threats Catalogue [5]. ISKE offers three security levels: low, medium and high. Every level has its own set of security measures. To achieve the required level of security, all required measures must be applied, described by the type and security objectives of the information assets. The centric position of ISKE is the confidentiality, integrity and availability of information what can be divided into three security levels (low, medium and high) [3].

1.2 Motivation

During the authors professional carrier at state institutions in the domain of information technology management and software development we have had real situations where higher information security requirement were needed. Compared to the private sector, state institutions are collecting and processing information for providing high level services, filling state needs or even fulfilling international contracts. Information security becomes more and more crucial in the world where more everyday life and operations depend on information technology and digital information. Moreover, we have participated in IT system developments, what should be addressed for information with medium or high (two highest) security levels and thus had to implement most security measures described by ISKE.

In most of those cases the (a) system must have data tampering detection, (b) data must be accessed rapidly, (c) data changes must be tracked for later supervision or analyse, (d) data might have to carry digital signatures, (e) data objects can have several entity versions, (f) systems must be flexible and simply maintainable and finally (g) data must be easily portable for future developments or migrations.

According to our professional understanding and informed awareness, the problem is that the author has not seen an info technology system that fully meets all information integrity requirements specified in ISKE and listed above. In Estonia there are currently different custom made systems for state registers and those implement some of the listed properties, but according to the authors professional awareness most solutions have some deep problems with one or another above listed characteristics. There is also a lack of publicly available discussion, resources and examples of implementing security controls in question.

The author of this thesis has maintained or taken part in projects where new information technology system for the state processed information developed or old systems rewritten. The above listed requirements and ISKE requirements are mentioned in the task, but none of the systems tat are developed by a third party partner, not fulfil all the security objectives of information assets. One cause can be the lack of references, experience or know-how in this specific domain.

For the security of information systems and reputation of institutions, this thesis will not mention any real cases or names. We only focus on possible comprehensive solution by positive examples.

1.3 Problem Statement

By analysis of the subject domain, providing theoretical solution design and implementing proof of concept, it is possible to help institutions and info technology developers to obtain system that meets security requirements required by ISKE. By publishing thesis and potential solutions as proof of concept, it can rise the awareness of ISKE application.

Main issues addressed in this thesis are:

- The lack of public and systematic approach to the subject domain including the lack of academic research;
- The lack of understanding about selected security requirements for data with high integrity needs according to ISKE;
- The lack of publicly available discussion and sample implementations for applying ISKE measures to info-system handling data with highest integrity needs;
- The lack of understanding what will accompany by the implementation of ISKE measures in integrity domain;
- The lack of understanding the data tampering detection process.

Research questions and corresponding research's sub-questions of this thesis are the following:

RQ1. - How should the proof of concept be designed to obtain security objectives of data with highest integrity need demanded by ISKE?

- RSQ 1.1. What is the organisational context and the assets of possible system and proof of concept implemented in this thesis?
- RSQ 1.2. What are the security risks to data with highest integrity need according to ISKE?
- RSQ 1.3. What is the risk treatment decision for reference information system and developed proof of concept presented in this thesis?
- RSQ 1.4. What are the security requirements for implementing solution and proof of concept to obtain security objective of data with highest integrity need demanded by ISKE?
- RSQ 1.5. What are the functional requirements for implementing solution and proof of concept.
- RSQ 1.6. What are the possible limitations and effects that accompany the implementation of requirements for data with highest integrity need demanded by ISKE?

RQ2. - How should the proof of concept and the system implemented to obtain security objectives of data with highest integrity need demanded by ISKE?

- RSQ 2.1. What is the possible architecture and system design of solution and proof of concept to obtain security objectives of data with highest integrity need demanded by ISKE?
- RSQ 2.2. What is the possible database structure for solution and proof of concept to implement requirements for data with highest integrity need demanded by ISKE?
- RSQ 2.3. What activities should possible database tampering detection processes involve.

RQ3. - How should the solution validation be performed to ensure the fulfilment of security requirements?

- RSQ 3.1. What method should be used to perform validation of implemented solution?
- RSQ 3.2. What are the implemented proof of concept performance characteristics?
- RSQ 3.3. Does the proof of concept fulfil the security and functional requirements demanded by this research?

1.4 Methodology and Scope

This section describes the methodology, that has been used during the research of this thesis and research work. Overall processes and methodology steps are shown on the figure (see Figure 1).

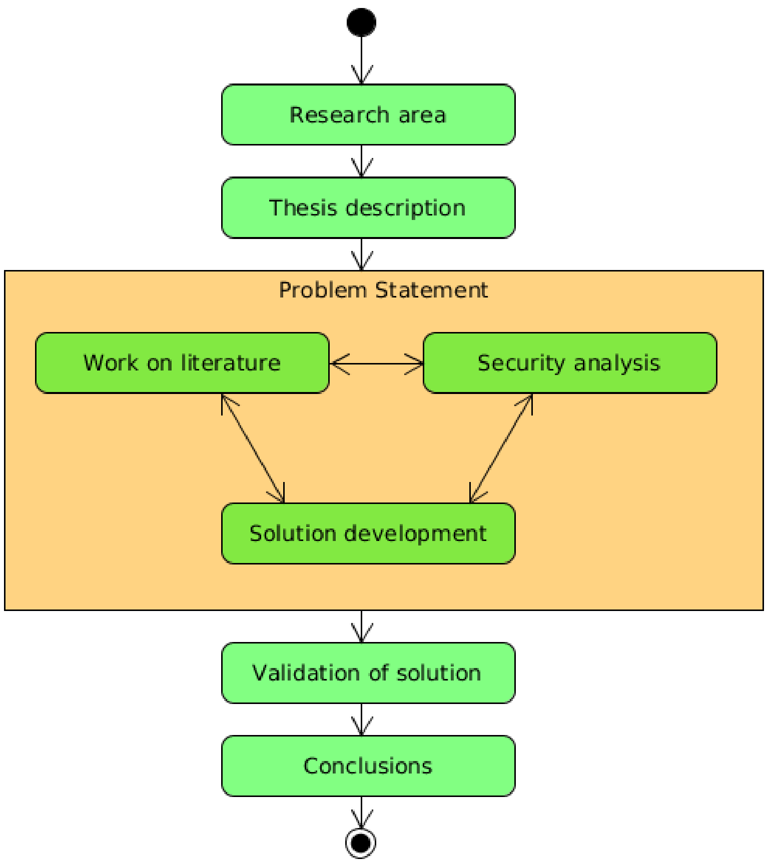


Figure 1: Research methodology process

Research starts with specifying the research area what is described as background in section 1.1, motivation of thesis in section 1.2 and problem statement in section 1.3. The research continues with working on the research questions and problem statements, discussions, security analysis and development of proof of concept. Also the solution are validated and the process ends with drawing conclusions.

In this section we also describe the scope of the research project. We define that in this thesis and during the research work conjunction with ISKE we focus only on the information integrity aspect of the highest security level of IKSE integrity subclass "T3". On the other hand we wish to develop a solution which won't only fulfil ISKE's measures in integrity domain, but will also help to answer all research questions listed in section 1.3.

Security analysis are done using Information System Security Risk Management (ISSRM) [6] methodology conjunction with ISKE Reference Guide [3] and ISKE Measures Catalogue [4].

The subject of the security analyse and proof of concept solution is based on a fictional state registry what in many ways mimics possible real state information inventory, including the integrity.

1.5 Outline

The thesis is structured as follows:

- Section 1: This section presents the Estonian security standard ISKE, the description of this thesis, problem statement, research questions, methodology and the scope of the research.
- Section 2: This section presents the security analysis according to the methodology of this research and scope of this thesis conjunction with ISKE.
- Section 3: This section presents a detailed architecture, discussion about the design and implementation of solution components.
- Section 4: This section presents the validation results of the developed proof of concept.
- Section 5: This section discusses about how to apply the solution.
- Section 6: This section summarises the results of the research.

2 Elicitation of Security and Functional Requirements

This section focuses on answering the research question RQ1. The sub-questions will be answered in te following subsections. The security analysis is bases on Information System Security Risk Management (ISSRM) [6] methodology and the input data is collected form ISKE. According to [7] the ISSRM process consists of six steps:

1. a study of the organisations context and the identification of its assets;
2. determination of security objectives;
3. risk analysis, that elicits which risks are harming assets and threatening security objectives;
4. risk treatment decision;
5. determination of a security solution (security requirements) to mitigate the risks;
6. security controls instantiation.

This section describes organisational context, its assets study and determination of security objectives in section 2.1, risk analysis in section 2.2, functional requirements in section 2.4.2, security requirements in section 2.4.1, possible limitations and effects of applying requirements in section 2.4.3. Security controls are discussed in section 3. In this section we have different diagrams visualising the concepts we describe in the subsections. Those diagrams are depicted as Graphical Misuse Cases diagrams [8].

2.1 Organisational Context, Assets Identification and Defining Security Objectives

In this section we will answer research sub-question RSQ 1.1.

2.1.1 Organisational Context

In this thesis we consider the system to be a hypothetical Estonian state registry for personal identity management, hereinafter Estonian Identity Registry what is established by the state act. This registry is created for fulfilling state needs and operating public processes. It is one of the most critical data inventories, cause other state processes rely on this registry. It is the base of the state guarantee of a persons virtual identity. This registry's data is stored permanently. All data collected into this inventory is specified by its Statute For Maintenance Of Inventory. In this hypothetical act, there is a specified security level "H" (High) according to rules specified in the act [1]. Security components class for integrity is specified as "T3" (highest). From the fact that this registry has quite a high usage load the availability component "T" is also important, as well as confidentiality component "S", but in this thesis we do not focus on the availability and confidentiality aspects. Because the overall security level is already driven by the highest value from integrity component. Information collected into the Estonian Identity Registry does not have restrictions corresponding to state secret. Estonian Identity Registry has a chief and authorised processor, what is a state institution compiled of a officials (business and IT-technical).

2.1.2 Assets and Security Objectives Related Context

One of the security aspects that ISSRM specifies are Assets [6]. Assets are divided into Business assets and Information System assets. For the system in question to this thesis we specified assets be the following:

Business Assets :

- personal identification data.

Information System assets :

- server (any kind) for storing or processing personal identification data,
- network for interchanging personal identification data;
- Database Management System to store identification data,
- institutions personnel including data managers, IT-administrators, chief of security, etc;
- third party state institution;
- subjects of personal identification data.

ISSRM also specifies the aspect of security objectives of information system by defining security criterion [6] on business assets. In our case for the system Estonian Identity Registry, the security objective is the **integrity of personal identification data**. Organisational, asset and security objective related concepts are shown on the figure below (see Figure 2). UML use-case actors represent participants who are interchanging or might interchange with system. UML use-cases represents the set of actions performed by a system. UML use-case painted in green and with stereotype "security criterion" represents he security objective.

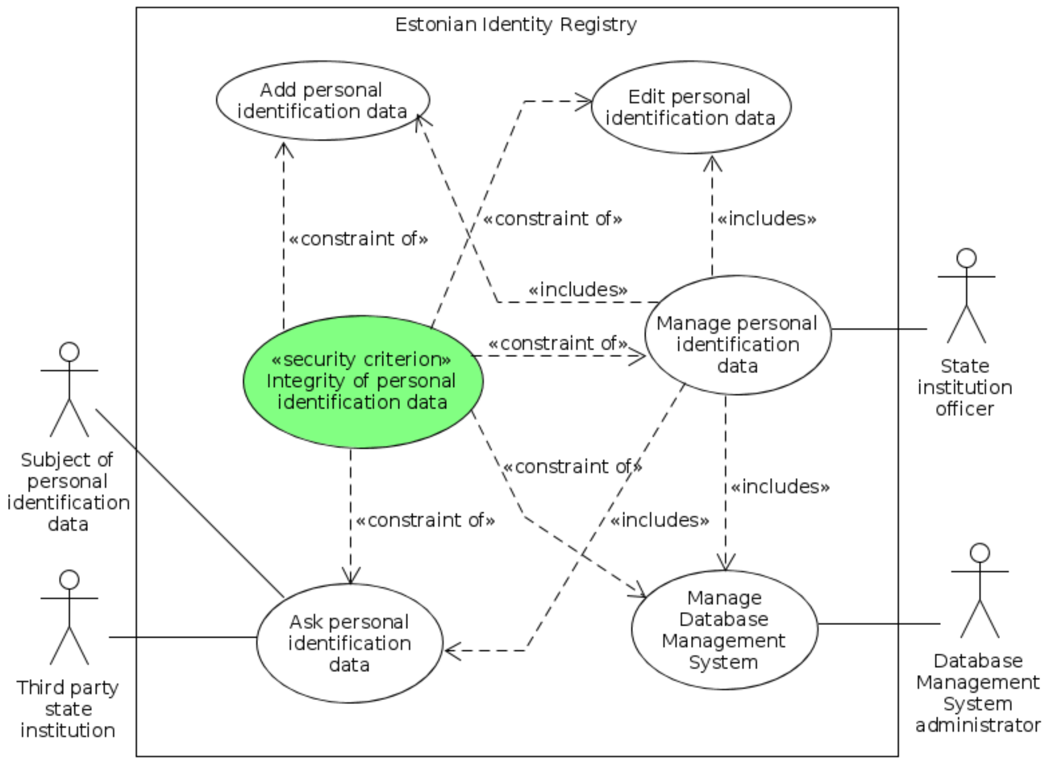

Estonian Identity Registry
Add personal identification data
Edit personal identification data
«constraint of»
«constraint of»
«includes»
«includes»
Manage personal identification data
«security criterion» Integrity of personal identification data
«constraint of»
«constraint of»
«constraint of»
«includes»
«includes»
Ask personal identification data
Manage Database Management System
Subject of personal identification data
Third party state institution
State institution officer
Database Management System administrator

Figure 2: Misuse case model about assets concept

2.2 Security Risk Related Concepts

In this section we will answer research sub-question RSQ 1.2. The detailed answer is provided in the following subsections.

2.2.1 Background

ISKE describes several risks and threats in ISKE Threats Catalogue [4]. Threats are divided into five categories:

1. G1: Force majeure;
2. G2: Organisational deficiencies;
3. G3: Human errors;
4. G4: Technical malfunctions and defects;

5. G5: Attacks.

If we look at those threats listed in ISKE Threats Catalogue in ISSRM context, then in detail those are not directly threats, but can be understood as a collection of standard risks. The name of threat in ISKE Threats catalogue represents potential impact in ISSRM context. Threats detailed description represents a set of collected theoretical aspects of: impact, vulnerability, threat, threat agent and attack method defined by ISSRM. To fully follow methodology of ISSRM we need to extract info from each risk topic.

We analysed the catalogue to find appropriate risks that suits in technical term the data integrity domain in software or database technical term. We did not took of any records from category G1, G2 and G3. Also threats that fall info configuration domain, because for example Database Management System configuration issues are not the scope of this thesis. Threats that ISKE defines to be hazarding for data integrity and one that considered important for this thesis are the following:

1. G4.13 Stored data loss (see section 2.2.3);
2. G4.28 Database data loss (see section 2.2.4);
3. G4.30 Loss of database integrity and compatibility (see section 2.2.5);
4. G5.64 Manipulation of data or software in database systems (see section 2.2.6).

2.2.2 Description of Collected Risks

On figure 3 we show Graphical Misuse Cases diagram [8] about risk related concepts. The diagram is based on ISKE risk descriptions, and is discussed in more detail in the following subsections. UML use-case actors painted with black represent malicious actors or threat agents according to ISSRM. In our case for example technical errors, human errors, human attacker, etc. UML use-cases painted in black represents malicious activity and links between theirs relationships. According to ISSRM those activities combine event, threat and attack method which are used by threat agent. UML use-cases painted in grey and with stereotype "vulnerability" represent possible vulnerability our reference system Estonian Identity Registry might have. UML use-cases painted in grey and with stereotype "impact" represent a possible impact what might harm the Estonian Identity Registry information system assets and therefore negates security objective "Integrity of personal Identification data" presented in section 2.1.2. UML use-cases with regular representation symbolise activities defined also in asset concepts in section and present the normal usage of the reference system. Links between UML use-cases represent relationships between concepts described in research [9].

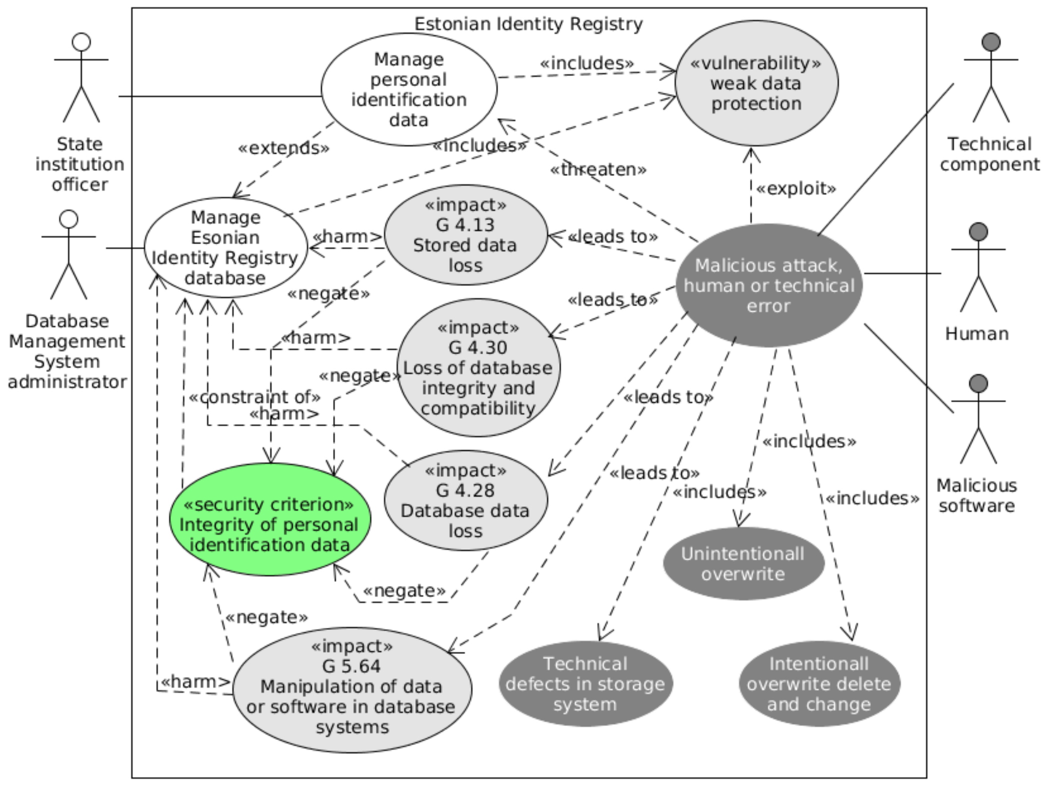

Estonian Identity Registry
State institution officer
Database Management System administrator
Technical component
Human
Malicious software
Manage personal identification data
«vulnerability» weak data protection
Manage Esonian Identity Registry database
«impact» G 4.13 Stored data loss
Malicious attack, human or technical error
«impact» G 4.30 Loss of database integrity and compatibility
«impact» G 4.28 Database data loss
«security criterion» Integrity of personal identification data
Unintentionall overwrite
Technical defects in storage system
Intentionall overwrite delete and change
«impact» G 5.64 Manipulation of data or software in database systems
«includes»
«extends»
«threaten»
«exploit»
«harm»
«leads to»
«negate»
«constraint of»

Figure 3: Misuse case model about risk concepts

2.2.3 G 4.13 Stored data loss

According to the description of the threat [10], loss of data or forgery could suppress or even stop the management or operation processes of authorities. It can harm trustworthiness of the institutions or management operations. Private companies or people can loose money or reliability. Destruction of data can disrupt every institutions every day operations or even stop them totally. Threat description provides several possibilities of data destruction. The most important of witch suit the context of thesis are listed below:

1. Data can be deleted or overwritten unintentionally;

2. Technical defects can occur in storage systems for example head crashes on hard drives;

3. Stored data can be added, changed or deliberately deleted by person or by malware.

2.2.4 G 4.28 Database data loss

According to the description of the threat [11] data losses can happen because of different reasons. For example data can be manipulated unintentionally and can be can be disappear caused by database errors or by a premeditated attack. Data loss can occur because of memory leakages in database management programs or in server and can be caused by a denial of service scenarios. But for whatever reason he data is lost, accessibility and integrity will suffer because of that. Problems like this can lead to similar consequences as discussed by "G 4.30 Loss of database integrity and compatibility" in section 2.2.5.

2.2.5 G 4.30 Loss of database integrity and compatibility

According to the description of the threat [12] loss of database integrity and compatibility is a situation where a database exists but there are errors. That is a situation where, data can not be accessed or processed correctly. Compatibility loss can occur in many ways beginning from unintended change and faulty synchronisation operations to premeditated attacks. It can lead to the following consequences:

1. Applications that need correct data in operations are influenced or won't work at all [12];
2. Available data reflects fake reality [12];
3. Database and its data integrity assessment and recovery takes much effort [12].

It can be impossible or hard to restore data or to identify reasons of loss of integrity and compatibility if measures have not contributed.

2.2.6 G 5.64 Manipulation of data or software in database systems

According to the description of the threat [13] manipulation of data or software is a situation where data or software is amended to be false or unsuitable for use. Possible consequences are the same as described by "G 4.28 Database data loss" in section 2.2.4 and by "G 4.30 Loss of database integrity and compatibility" discussed in section 2.2.5.

2.3 Risk Treatment and Security Requirements

This section describes risk treatment-related and requirements-related concepts of ISKE and and analysis about those requirements. Research sub-question RSQ 1.3. and RSQ 1.4 will be answered.

Analyse is done according to ISSRM methodology on risk treatment-related concepts [6]. For illustrating those concepts we have used misuse case modelling technique specified in research [9].

2.3.1 Risk Treatment-related Concepts

In ISSRM risk treatment is known as the decision of how to treat the identified risks [6]. This decision can lead to security requirements. According to ISSRM [6] risk treatment decisions fall into the following categories:

- risk avoidance decision - actions will be taken to fully avoid the risk;
- risk reduction decision - this includes actions to reduce the probability or negative consequences or both connected to the risk;
- risk transfer decision - a third party is involved in the process of having the burden of loss from a risk;
- risk retention decision - is accepting the risk and its consequences.

ISKE also has a concept of risk treatment. ISKE Reference Guide [3] and in Measures Catalogue [4] guides how decisions must be made in different conditions. Different risks can be treated differently.

In the case of our reference system Estonian Identity Registry we have the security objective **integrity of personal identification data** as presented in section 2.1.2. According to the organisational context description in section 2.1.1 our risk treatment decision can be **risk avoidance** or **risk reduction** [6]. According to a widely known fact, there is no such thing as full security and thus there is probably no way to fully avoid data integrity loss. Even if data is written on paper it's possible to tamper it, data can get loss in accidents or etc. If managing is done in an off-line info system there will be no benefit from that for the general public. Even cutting info system out of a public network there can be malicious insider who tampers data. According to our concept we consider risk treatment decision to be a **risk reduction decision** [6].

2.3.2 Security Requirements-related Concepts

ISSRM defines security requirement as "a condition over the phenomena of the environment that we wish to make true by installing the IS, in order to mitigate risks" [6]. Security requirements specify actions that need to be taken, to reach the situation defined by the security objective. ISSRM also defines security controls as "a designed means to improve security, specified by a security requirement and implemented to comply with it" [6]. Controls are derived from security requirements and implementation description of controls are discussed in next main section 3.

In this thesis security requirements are collected from ISKE Measures Catalogue [4] items, hereinafter measures. The selection is made from the section what applies to security component sub class for integrity "T3" specified by organisational context in section 2.1.1. In our point of view, the particular catalogue consists of a confusing mixture of requirements and possible security controls. In some cases the description even states the required implementation or product to use. Our arguments are based on [14] where the specification of good requirements are specified. Nonetheless, in following subsections, a description of particular measures are given to provide background and a better understanding about the means of particular measure.

The measures are considered as key measures in this thesis are:

1. "HT.10 Database records cryptographic linking" [15] (see section 2.3.3);
2. "HT.34 Usage of digital signature" [16] (see section 2.3.4);
3. "HT.52 Additional requirements to cryptographic tools" [17] (see section 2.3.5).

2.3.3 HT.10 Database records cryptographic linking

According to this measure description [15], this security measure requires to use a cryptographical linking method taking into account the following principles:

1. Entries of the database must be linked in chronological order with the cryptographic chain (local time stamp) and cryptographic hash digest preimage resistance property must be used;
2. Used chaining technique must prevent unnoticeable data deletion;
3. Used cryptography tools must comply with measure HT.52 described in section 2.3.5;
4. The defined employee must be committed to a periodical cryptographical link inspection. This requirement must be documented in a persons contract of employment, job description or notes.

2.3.4 HT.34 Usage of digital signature

According to this measure description [16], this security measure includes rules and guidelines for digital signature usage:

1. With digital signature and/or with digital seal what meets the requirements provided in the Estonian Digital Signatures Act is allowed to equip both databases, records, fields, or entire database tables, or entire databases, etc. Every authority must attaches itself to assess the evidential value of its own data and, consequently, must evaluate whether to use the personal digital signature or digital seal.

2. Data and documents with security sub-class T3 is prohibited to use digital signature mechanisms, which do not meet (especially in infrastructure issues) the Estonian Digital Signatures Act requirements (that are classified, such as PGP and GnuPG signatures).

3. Technical and supportive data can be left digitally unsigned. If a database or info system main data is wanted to be left digitally unsigned, then several security zones in mean of ISKE must be created and the unsigned data most be taken into a lower zone.

NB! ISKE measure text is older that the Estonian Parliament decree, that Digital Signatures Act is repealed and substituted by new act "Electronic Identification and Trust Services for Electronic Transactions Act" [18]. It substantially changes the scope of allowed digital signature types and formats compared to the Digital Signatures Act.

2.3.5 HT.52 Additional requirements to cryptographic tools

According to this measure description [17], this security measure includes following guidelines:

1. Hash functions MD2, MD4, and MD5 usage is (without exception) prohibited, regardless of their place of use, uses and usage;

2. Cryptographical and/or -protocol must be disclosed or made available at least two years ago for cryptoanalysts to ensure it's security. Closed crypto-tool must be passed through security audit where unbreakability is convinced. In that case, head of the security authority must ensure reliability of such security audit and accept it.

3. If the cryptographic digest used for proof of value (integrity) has to ensure more than ten years, hash function with output (hash length) at least 256 bits must be used. Then the usage of the hash function RIPEMD160 and SHA1 are forbidden. It is advisable to use the SHA2 family hash functions or with longer hash values RIPEMD variants.

4. If the Crypto algorithm with a public key is used in conjunction with a digital signature, that must ensure proof of value more than then years, then the RSA key length must be at least 1536 bits. Using 1024 bit RSA usage is prohibited in those cases.

5. If the Crypto algorithm with a pubic key is used in conjunction with a digital signature, that must ensure proof of value more than fifteen years, then the RSA key length must be at least 4096 bits.

2.4 Requirement Elicitation and Specification For Solution

In this section we will answer research sub-questions RSQ 1.4., RSQ 1.5. and RSQ 1.6. We provide our interpretation of security requirements based on data collected from ISKE security measure descriptions mentioned in previous subsections. The interpretation of requirements and formulation is based on the understanding and convictions of the author and techniques presented in [14]. In addition we provide additional functional requirements what proof of concept must fulfil, to act as information system.

2.4.1 Security Requirements For a Solution

Security requirements for solution are the following:

SECREQ.1	The solution must support main data signing with a digital signature.
SECREQ.2	A digital signature must be in format, that complies with the Electronic Identification and Trust Services for Electronic Transactions Act.
SECREQ.3	A digital signature must be based on a public key cryptography.
SECREQ.4	A digital signatures public RSA key length must be at least 1536 bits.
SECREQ.5	The solutions database main data table records must be cryptographically linked.
SECREQ.6	The solutions database main data table records must be linked in chronological order.
SECREQ.7	The solutions database main data table cryptographical link must use hash digest preimage resistance property.
SECREQ.8	A cryptographical links hash digest must be at least 256 bits.
SECREQ.9	The solution must support the databases main data tables full tampering detection.
SECREQ.10	The solution must support databases main data tables partial tampering detection.
SECREQ.11	The solution must support databases main data tables partial tampering detection with tampered database record detection.
SECREQ.12	The cryptographical link meta data needed for data tampering detection must be stored separately form secured data. (Derived from the need discussed in section 2.4.3)

In figure 4 we used Graphical Misuse Cases diagram [8] to visualise security requirements related to our reference system Estonian Identity Registry and developed proof of concept. UML use-case cases with stereotype "security requirement" represent the security requirements (security use case) presented in current section, which mitigates the identified threats covered in the section 2.2 drawn as misuse cases (painted in black).

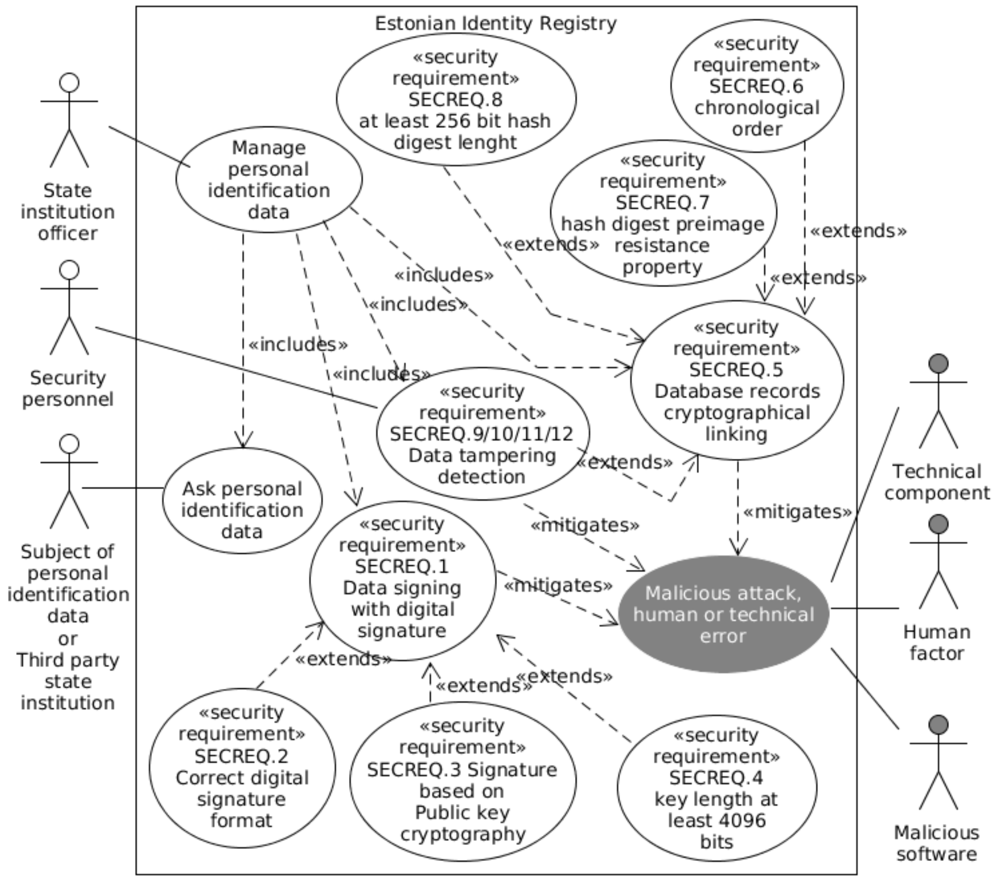

Estonian Identity Registry
State institution officer
Security personnel
Subject of personal identification data or Third party state institution
Technical component
Human factor
Malicious software
Manage personal identification data
Ask personal identification data
«security requirement» SECREQ.8 at least 256 bit hash digest lenght
«security requirement» SECREQ.6 chronological order
«security requirement» SECREQ.7 hash digest preimage resistance property
«security requirement» SECREQ.5 Database records cryptographical linking
«security requirement» SECREQ.9/10/11/12 Data tampering detection
«security requirement» SECREQ.1 Data signing with digital signature
«security requirement» SECREQ.2 Correct digital signature format
«security requirement» SECREQ.3 Signature based on Public key cryptography
«security requirement» SECREQ.4 key length at least 4096 bits
Malicious attack, human or technical error
«includes»
«extends»
«mitigates»

Figure 4: A misuse case model about risk treatment concepts and security requirements

2.4.2 Functional Requirements Of a Solution

Functional requirements of a solution are the following:

FUNREQ.1 A solution must provide a standardised application programming interface (API);

FUNREQ.2	A solution API must support the main data input operation.
FUNREQ.3	A solution API must support the main data output operation.
FUNREQ.4	A solution must support validating digital signatures of signed main data during data output operations.
FUNREQ.5	A solution must support the main data storing to relational database.
FUNREQ.6	A solution API must support database tampering detection executing operations.
FUNREQ.7	A solution API must support database tampering detection result output operations.
FUNREQ.8	A solution must support database tampering detection result storing into a database.

2.4.3 Limitations and Effects That Accompany With Implementing Security Requirements

In this section we discuss possible limitations and effects of security requirements to practical implementation.

Based on ISKE measures and elicited requirements specified in sections above we conclude following aspect and effects.

Firstly we suggest that if an info-system asset database server or database management system software is threatened, then all artefacts in the system are threatened. In that case chronologically ordered cryptographic hash link, the backbone of this security solution, is also threatened equally with the data that's security we try to achieve. In this case, the protection of the last cryptographic hash function digest is very important. If the last database row cryptographic link digest is not carefully protected, then attacker can change data, manipulate with hash values by generating a new link or add data to the last database record without notice. Last database records can be changed or deleted by new records without being able to identify the fact that tampering has been comited. We state that for later data tampering detection, it is important to deposit a cryptographic link record and other important meta-info about database entry to another system. This system must be under control of a different department or personnel of an operation institution. In this case, last digest of cryptographic link is also deposited out of the database and the link validity check will fail if hashes from the period under validation won't mach. System where hashes will be deposited should be very dedicated to this function, carefully configured, maintained and operated.

The second important limitation of chronologically ordered cryptographic link is that it is not possible to normally change database records without recalculation of its hash digest and the following link. Such calculations can be time consuming and growing volume of database records will also grow the time for calculations. Parallel calculation is also not possible, so every change will depend on the previous and must wait until the link recalculation of the last record change is done. In case of aappend-only database technique as described in the thesis [19] and in proceeding [20] can be considered as a practical implementation pattern and a logical change and deletion of data object can be possible. This approach on the other hand can be effective by means of data history, because otherwise a database model should be designed to allow historical view of changed data objects.

Thirdly what need to be pointed out is associated with usage of digital signature required by security requirement. Security requirements SECREQ.1, SECREQ.2 and SECREQ.3 listed in section 2.4.1 can be practically implemented in Estonia by using services provided by the Certification authority (legal name SK ID Solutions AS) [21] and a toolkit provided by the Information System Authority Estonia. There is no other option at the moment in Estonia to achieve mentioned requirements.

2.5 Summary

In this section we answered the research question RQ1 and its sub-questions. Through analyse steps we researched and discussed aspects of organisational context, assets and security objectives of the reference system and developed solution. We analysed and presented collected security risks based on ISKE threats collection and proposed the risk treatment concept and security requirements elicited according to research on ISKE security measures. Finally we proposed solutions for functional requirements and discussed limitations and effects of implementing of security requirements.

3 Proof of Concept for ISKE selected Integrity Requirements

This section describes the solutions technical architecture and its components. The solutions proof of concept is built to show hot to meet security and functional requirements. The solution applies security requirement by implementing security controls. In this section we will answer research question RQ2 stated in section 1.3 and by research sub-questions in subsections.

3.1 The Solution Architecture and Components

In this section we will answer research sub-question RSQ 2.1. The basic architecture of solution we provide is shown below (see Figure 5). The solution architecture is divided into subsystems (painted with green colour on drawing). Each subsystem is discussed in subsections. Additionally, there is a third party component (painted with yellow colour on drawing), what is not developed during the research of this thesis, but is used by proof of concept and a solution to help implement security controls. We will briefly describe role of this counterpart. In each subsystem, there are hosted a number of applications and artefacts that all together form a working solution and proof of concept.

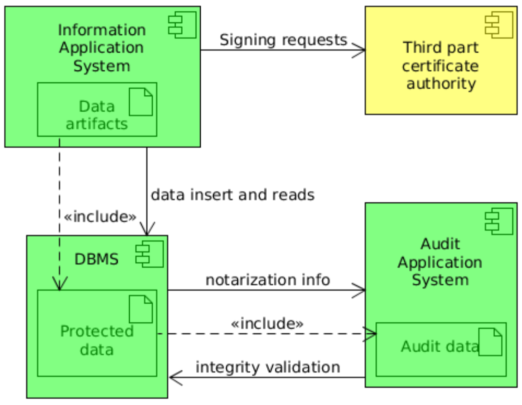


Figure 5: The basic architecture of a solution

Overall the architecture consists of elements listed below:

1. Information Application System (see section 3.2);
2. Database Management System (see section 3.3);
3. Audit Application System (see section 3.4);

4. External third party trustee (see section 3.6).

Each subsystem plays a role in implementing one or more security or functional requirements. A more detailed architecture is described on solutions UML deployment diagram (see Figure 6). This diagram describes the detailed physical deployment of a solution including servers, execution environments, artefacts, communication paths and protocols between subsystems.

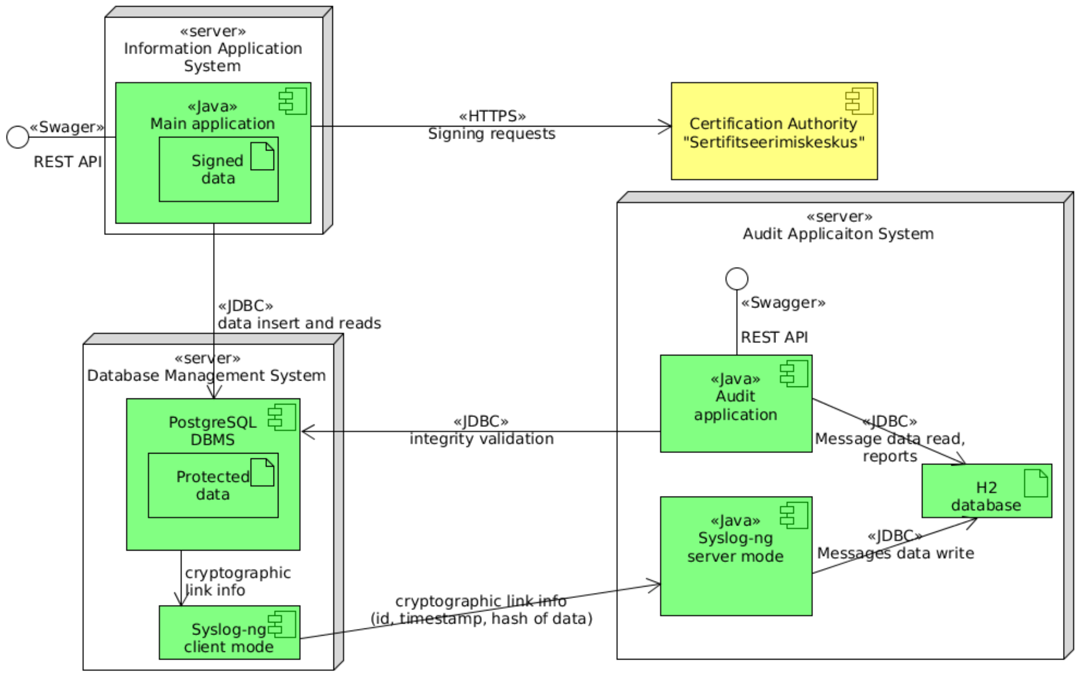


Figure 6: Solution deployment diagram

3.2 Information Application System

The role of the Information Application System is to act as the main data manager. It is main switch between possible client applications using REST API, external participants web services and the Database Management System (DBMS) where all data is stored. The information Application System is developed in Java programming technology using Spring Boot Framework [22]. The role of the Information Application System is to:

1. Provide RESTfull services to clients for creating and accessing data - fulfils functional requirement FUNREQ.1, FUNREQ.2 and FUNREQ.3;

2. Manage data signing with digital signatures - fulfils security requirement SECREQ.1, SECREQ.2 and SECREQ.3;
3. Evaluate correctness and validity of digital signatures - fulfils functional requirement FUNREQ.4;
4. Communicating with Database Management System for data storing - fulfils functional requirement FUNREQ.5.

3.2.1 Information Application System API description

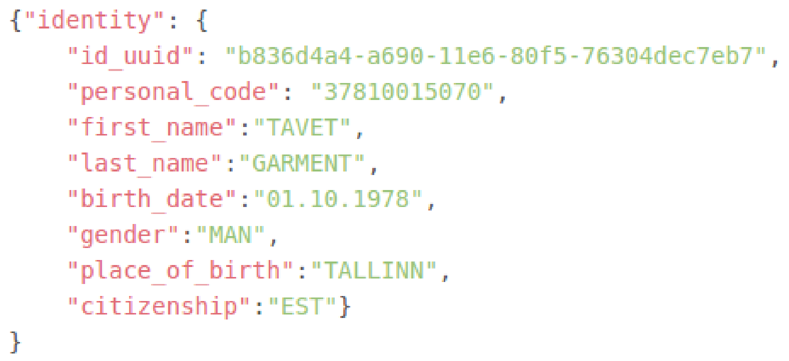
As described in the best practices document [23] RESTfull services are services implemented using REST (Representational State Transfer) architectural style [24]. Information Application System application programming interface (API) consists of one REST Controller with following endpoints:

- **/getIdentity** - HTTP GET request for getting specific Identity data object according to its UUID (universally unique identifier).
- **/postIdentity** - HTTP POST request to add new Identity data object.

GET and POST requests returns data formatted in JSON. For visualising, describing and documenting provided API we integrated REST API framework Swagger [25] into Information Application System. It automatically generates documentation and basic user interface for services provided by the application.

3.2.2 Data Format For Main Data Storing

For storing main data we considered XML or JSON. Those formats are compared in articles [26] and [27]. According to mentioned studies JSON has better performance characteristics than XML. Although, cited articles both acknowledge the fact that both formats might provide unique strengths and both have properties that demonstrate the strengths and weaknesses relative to each format. But rather important is how it is possible to natively integrate those data formats into Database Management System. To look at the most serious and wide spread Database Management Systems created for example by Microsoft [28], Oracle [29], IBM [30], PostgreSQL Global Development Group [31] [32], those all have both XML and JSON native support in some form. But according to cited references we understand that the use of JSON is extensively spreading and became the most popular data format in many use cases. Also we can see that JSON support have more proactively be developed in those systems in the future than XML. Therefore in our solution, we propose to use JSON data format for storing data as machine readable format in the database system. On figure 7 the sample data object, what consists of personal identity data, is presented.



```
{"identity": {
    "id_uuid": "b836d4a4-a690-11e6-80f5-76304dec7eb7",
    "personal_code": "37810015070",
    "first_name":"TAVET",
    "last_name":"GARMENT",
    "birth_date":"01.10.1978",
    "gender":"MAN",
    "place_of_birth":"TALLINN",
    "citizenship":"EST"}
}
```

Figure 7: Identity data object formatted in JSON

3.2.3 Digital Signature Implementation

The management of digital signatures is a crucial to this solution. By digitally signing data and documents it is possible to gain a long term data integrity guarantee. Estonian digital signature solution came into effect on 7th October 2002 when the first official digital signatures was given using the "DigiDoc Client" [33] program. So digital signature framework has been already utilised for 14 years in Estonia. In October 2016 a new act "Electronic Identification and Trust Services for Electronic Transactions Act" [18] came into force, what made a major change into our digital signatures field by repealing the old Estonian centric digital signature policy and taking into effect the European Union regulation about Trust Services and Electronic identification. With that, unified digital signature types and signature containers formats where introduced what are compliant all over Europe.

We implemented a digital signature management using "Digidoc4j library" [34] provided by the Estonian Information System Authority. This library wraps different functionalityes connected to digital signatures creation on validation. Common signature container formats [35] available for creation using the library are the following:

- **BDoc format** - Obsolete Estonian digital signatures with OSCP confirmation without time stamp or time mark;
- **BDoc 2.1 format or ASiC-E LT-TM** - Signatures with OCSP confirmation and time mark;
- **ASIC-E LT format** - Signatures with OCSP confirmation and time stamp for long time integrity guarantee.

In our solution, we use digital signature containers with **ASIC-E format** and digital signatures with **LT** signature profile. This container format has file mime-type **vnd.etsi.asic-e+zip** and extension **.asice**.

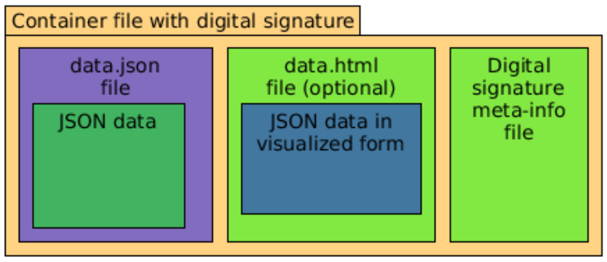

Container file with digital signature
data.json file
JSON data
data.html file (optional)
JSON data in visualized form
Digital signature meta-info file

Figure 8: Data encapsulation in digital signature container

On figure 8 we present how we encapsulate data into a digital signature container. After data is received by RESTfull services and processed, it is put into file. Additionally, there can be a secondary form of data for visualising data in human readable format for example as HTML file. Such data can be for example application or decision about application what need to be the processed by applicant or an officer of a state institution. In our solution we only implemented proof of concept with a machine readable file. After the data file generation is completed, is passed into signature module. In that module the signature container file will be formed, data files will be located in this container. After signing the digital signature container content, the signatures meta-info will be placed into the container and as well raw JSON data will be passed to the database.

On figure 9 we visualise flow of JSON data from source to Database Management System.

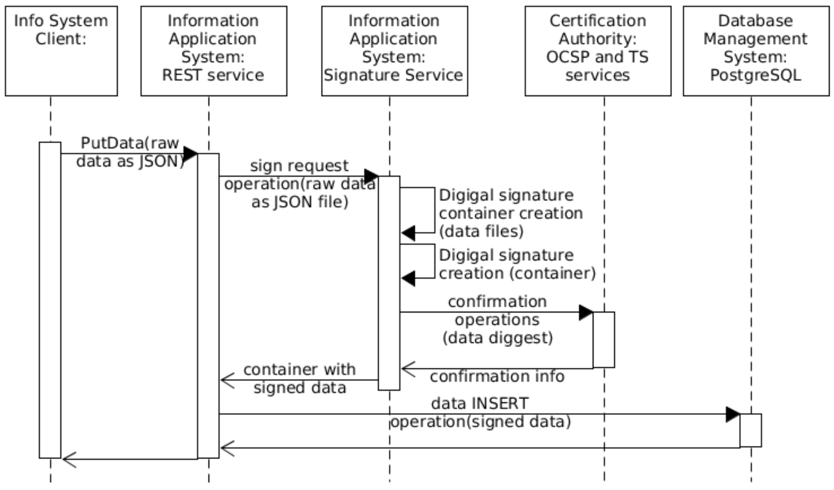

Info System Client:
Information Application System: REST service
Information Application System: Signature Service
Certification Authority: OCSP and TS services
Database Management System: PostgreSQL
PutData(raw data as JSON)
sign request operation(raw data as JSON file)
Digigal signature container creation (data files)
Digigal signature creation (container)
confirmation operations (data diggest)
confirmation info
container with signed data
data INSERT operation(signed data)

Figure 9: Flow of data from client to database including digital signing

If the client requests the data from RESTfull service, raw data and container with signed data will be requested from the Database Management System. After receive of the artefacts, the container will be passed into signature module that validates signatures using Certification Authority services. After signature validation, JSON data file will be extracted from container and retrieved to client.

On figure 10 we visualise flow of digitally signed data validation and retrieve to client.

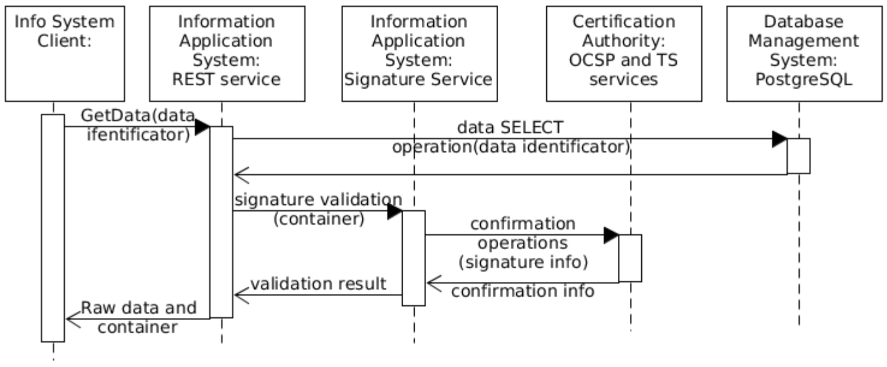

Info System Client:
Information Application System: REST service
Information Application System: Signature Service
Certification Authority: OCSP and TS services
Database Management System: PostgreSQL
GetData(data ifentificator)
data SELECT operation(data identificator)
signature validation (container)
confirmation operations (signature info)
confirmation info
validation result
Raw data and container

Figure 10: Flow of data validation during output data to the client

This part of application will fulfil security requirements SECREQ.1, SECREQ.2, SECREQ.3 and SECREQ.4.

3.3 Database Management System

In this section we will answer research sub-question RSQ 2.2. In our solution the Database Management System carries the central data storage role. It provides interface for storing and accessing data and organises cryptographic linking required by security requirements. In our solution we use the "PostgreSQL" [36] version 9.6.2 as DBMS. The Database Management System role is to:

1. Manage data storing and retrieving - fulfils functional requirement FUNREQ.5;
2. Organise cryptographic linking of database table entries - fulfils security requirement SECREQ.5-8;
3. Depositing cryptographic hash info to Audit Application System - fulfils functional security requirement SECREQ.9-11.
4. Calculating cryptographic links for correctness evaluation - fulfils functional security requirement SECREQ.9-11.

3.3.1 Data Storage Strategy

As a key technical decision for implementing solution was to use JSON data format for storing data we can classify our system as a document-oriented database [31]. For that type of storing solution NoSQL [37] has become a synonym. As refered before, document store using the JSON data format is classified as an NoSQL database. The NoSQL can be "No to SQL" but also as "Not Only SQL" [38]. According to the mentioned research there are many database management systems developed that aim to be NoSQL databases. Traditional Relational Database Management Systems are less mentioned in categories of NoSQL databases, but the last-mentioned systems benefiting more from NoSQLs ideology.

In our solution "PostreSQL" database management system has the ability to store JSON data format [32],[39] and used in conjunction with benefits of a relational database model. In PostgreSQL there are two possibilities to store JSON data a) using JSON data type, where data is stored as a string in the database and b) using JSONB data type, where data is stored in binary form in the database system. Since JSONB data type has been introduced in PostgreSQL the performance has been rapidly increased using GIN indexes [40]. By utilising this feature of PostgreSQL we managed to develop a highly performing solution without using the traditional approach of the Relational Database Management System.

3.3.2 Database Main Table Structure

For our proof of concept we developed a simplistic database structure with one database table. The database table named **main_data** stores data about persons identity for our reference system. The database table structure for **main_data** is described below:

- **id (bigint)** - PostgreSQL big integer type column for the primary key, indexed with b-tree index;
- **id_uuid (uuid)** - PostgreSQL UUID type column for an externally usable unique identificator, indexed with b-tree;
- **history_id_uuid (uuid)** - PostgreSQL UUID type column, identificator for versioning purposes, indexed with b-tree;
- **main_data (jsonb)** - PostgreSQL JSONB type column for data search, indexed with GIN index;
- **file (text)** - PostgreSQL text type column for storing digitaly signed data;
- **hashchain (character varying(255))** - PostgreSQL varchar type column 255 characters long for cryptographic linking purposes.

Column **file**, that is a text column type will store a BASE64 encoded digital signature container specified in section 3.2.3, what consist of digitally signed data files and a signatures meta-info.

The following database query example demonstrates how to select data from the database table designed by us containing a JSONB data filed that has a GIN index:

```
SELECT * FROM main_data WHERE main_data ->>
      'personal_code' = '37810015070';
```

In this sample we query the database for records in table "main_data" with a JSON key personal_code having value 37810015070 present in its structure.

3.3.3 Implementing Cryptographic Link

In this section we describe how we implemented security requirements SECREQ.5, SECREQ.6, SECREQ.7, and SECREQ.8. We give a brief overview about the database table row cryptographical linking.

In our solution, cryptographic linking is applied to database table rows using PostgreSQL DMBS database triggers and specially developed C-Language Functions [41]. If we insert a row into database table described in section 3.3.2, database trigger will be executed before inserting the new record. This trigger uses the function from the shared object library file developed using the C programming language. The prototype of those functions are described by the author in the thesis [42]. During the research part of our thesis project we have obtained and improved a source code of cryptographic operations discussed in the mentioned thesis.

The flow of data from the Information Application System to the database and the cryptographic link meta-data deposition to Audit Application System is graphically presented on figure 11.

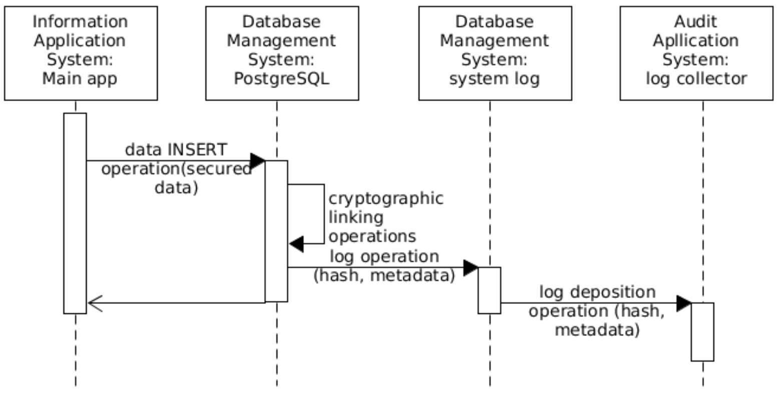

Information Application System: Main app
Database Management System: PostgreSQL
Database Management System: system log
Audit Apllication System: log collector
data INSERT operation(secured data)
cryptographic linking operations
log operation (hash, metadata)
log deposition operation (hash, metadata)

Figure 11: Process of cryptographic linking and link meta-data depositing

The process of cryptographic linking is described below and graphically presented on figure 12:

1. The database receives data to insert into a secured table;
2. The database executes trigger for the INSERT statement;
3. The database trigger executes a C-language function in the shared object library for linking procedure;
4. The function in the shared object library makes SELECT query to the same table to get hash from the last inserted row;
5. The function generates a SHA-256 [43] hash from the output of the last operation and the data need to be inserted into table;
6. The function saves data and the corresponding hash generated in the last operation into table;
7. The function outputs the newly generated hash and a row meta-data to the system logging software described in section 3.4.2.

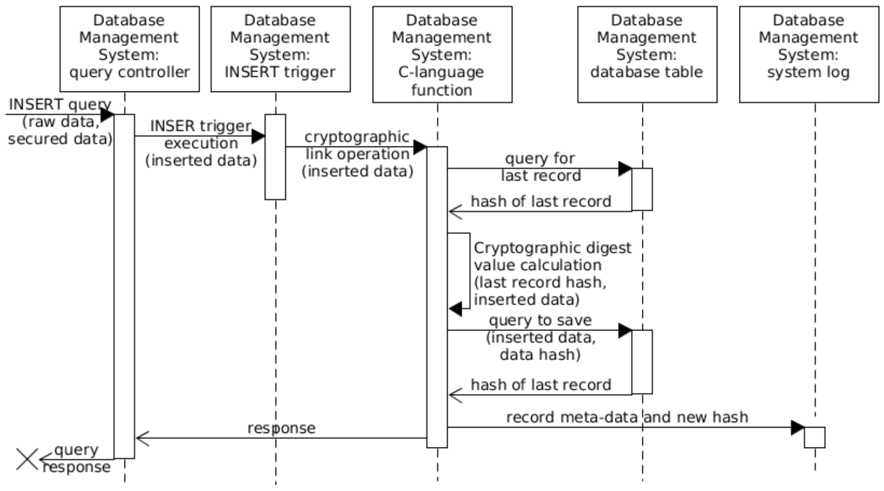

Database Management System: query controller
Database Management System: INSERT trigger
Database Management System: C-language function
Database Management System: database table
Database Management System: system log
INSERT query (raw data, secured data)
INSER trigger execution (inserted data)
cryptographic link operation (inserted data)
query for last record
hash of last record
Cryptographic digest value calculation (last record hash, inserted data)
query to save (inserted data, data hash)
hash of last record
record meta-data and new hash
response
query response

Figure 12: Detailed view of cryptographic linking flow

The source code of the mentioned discussed process is provided in appendix I. The cryptographic hash chain generation source code.
In Database Management System, "Sylog-ng" daemon is configured so that it sends all Syslog log messages to the remote Syslog-ng" log collector server what is deployed in the Audit Application System 3.4. Then we will send record meta-data and hash value for later tampering detection purposes. Mode information about selected "Syslog-ng" software is provided in section 3.4.2.

3.4 The Audit Application System

The third main subsystem of our solution is an Audit Application System. It is implemented to fulfil security requirements SECREQ.9-11 and functional requirements FUNREQ.6-8. This system collects cryptographic link hashes, meta-info about secured data and conducts database tampering detection activities. Audit Application System's role is to:

1. Collect cryptographic link information;
2. Store cryptographic link information;
3. Perform database tampering detection;
4. Store database tampering detection results.

As we discussed in section 2.4.3, to take advantage of a cryptographic link to data tampering detection, it is important to deposit the links info from the Database Management System to a more highly secured and trusted system what is controlled by a different organisational unit and person than Database Management System. In following sections we will describe the implementation details of a hash link data deposition and describe the proposed process for detecting database tampering.

3.4.1 Cryptographic Link Information Exchange Background

During the research part of this thesis we wanted to reuse available software, frameworks and solutions as much as possible. We did not want to develop new communication protocols or software that could increase the complexity of the solution and therefore the risk of failure. For depositing cryptographic link meta-data from the Database Management System to the Audit Application System we used Syslog [44] to collecting cryptographic link information. It is known fact that Syslog is widely used in networking and computing overall and in the software field there is lot of options to relay on. Also we decided to store link data in a light weight database file, what can simply be integrated to the Srping Boot framework. Options for this database file are for example Sqlite3, H2 or HSQL databases.

3.4.2 Implementating the Cryptographic Link Information Exchange

We researched softwares "Rsyslog" [45] and the "Syslog-ng" [46]. Overall it turned out, that possible fulfilling objectives using "Rsyslog" was not possible in a reasonable time. During the research of this thesis we deeply investigated different possibilities to transport and mediate Syslog messages using the abve mentioned software. By default a "Rsyslog" saves messages to text file. There is also "Rsyslog" module called

"Omprog" [47] what enables to forward messages to a specially developed program written programs in programming languages Java or Python. Therefore those programs can offer custom implementation forwarding a message. There is also module "Omlibdbi" [48] what is SQL database integration module, to forward messages to several databases. We developed programs in both Java and Python but unfortunately we were not able to integrate "Rsyslog" reliable working integration with those programs. Documentation seamed to be outdated and the integration with developed programs was not flexible enough. Also we were not able to make the "Omlibdbi" module working with "Sqlite3" database that was an option mentioned before. It turned out that it was not possible due to the lack of library support or conflicting library versions on the system level, what could turn out to be problematic in the future, if it as put into real use.

Secondly we investigated the "Syslog-ng" software that seemed promising and more actively supported by developers than "Rsyslog". This software also supports a similar approach to "Rsyslog", that the Java or Python development language can use to expand functionality provided by "Syslog-ng". First problems raised just after installing "Syslog-ng". We realised that the mentioned approach was not activated by default. We needed to compile a "Syslog-ng" to support expansion programs in Java and Python. We faced different problems including errors during the base software installation and aspects not clearly documented in the documentation. We concluded that using the program implemented in Python was not possible at this time due to unclear errors. But we were able to activate and use the program implemented in Java. We converted the same program implemented during the test with "Rsyslog" to the Java module program supported by "Syslog-ng". Alternatively we were able to save messages into the "Sqlite3" database, that could be used as a second option for storing hash chain log messages.

For conclude to implement security requirement SECREQ.9 and SECREQ.10, SECREQ.11 and SECREQ.12 we used the "Syslog-ng" software. The Database Management System, there is "Syslog-ng" installed in client and in server mode located in the Audit Application System. In the Audit Application System "Syslog-ng" is configured to save cryptographic link information to the "H2" database using Java module program implemented during this research project. The "H2" database structure is described in section 3.5.2. The llow of log message is described on figure (See figure 13) and are the following:

1. "Syslog-n"g server configures the database Java module program;
2. The module connects to the "H2" database file.
3. When the "Syslog-ng" client on the Database Management System receives a Syslog message, it forwards the message to "Syslog-ng" server;
4. "Syslog-ng" server delegates the message to Java module program;

5. The database module program inserts message data with the SQL query to the "H2" database;

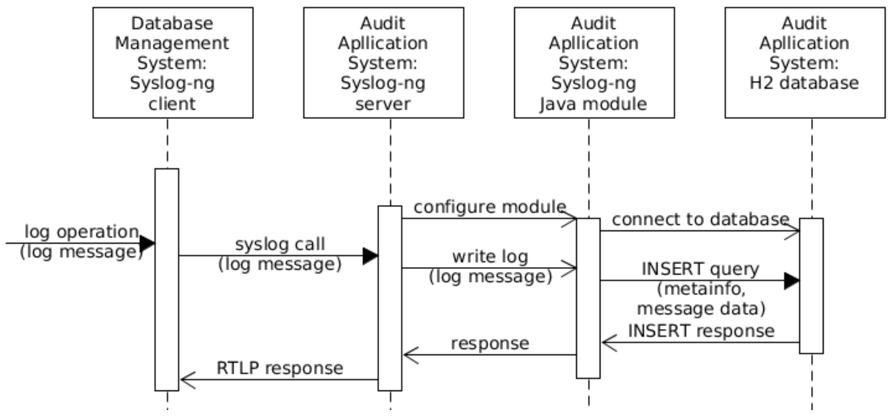

Database Management System: Syslog-ng client
Audit Apllication System: Syslog-ng server
Audit Apllication System: Syslog-ng Java module
Audit Apllication System: H2 database
log operation (log message)
syslog call (log message)
configure module
write log (log message)
connect to database
INSERT query (metainfo, message data)
INSERT response
response
RTLP response

Figure 13: Process of cryptographic linking and link meta-data depositing

The cryptographic link log message format is specified as follows:

```
YYYY-MM-DDTHH:MM:SS+HH:MM server_name program_name:
   <message text>
```

Message text consists of:

```
audit:<table_name>,<id>,<id_uuid>,<last_hash>,<new_hash>
```

Message transport assurance is an important feature in the cryptographic link exchange. It is important to guarantee of transport of every single log message transported from the Database Management System to Audit Application System. To lower the risk of loosing messages it is possible to use the Reliable Log Transfer Protocol [49] in the message transport. "The RLTP^{TM} protocol works on top of TCP, and can use STARTTLS for encryption. RLTP^{TM} supports IPv4 and IPv6 addresses. Inside the RLTP^{TM} message, the message can use any format, for example, RFC3164 (BSD-syslog) or RFC5424 (IETF-syslog)." [49]. This protocol is only available in "Syslog-ng Premium Edition" what is not available for free, so during the research project we did not implement transportation using the above mentioned protocol.

3.5 Database Tampering Detection Management Software

The role of the Database Tampering Detection Management Software is to manage, initiate and perform the validation of a cryptographic link in the Database Management

System described in section 3.3. This component is developed in the Java programming technology using Spring Boot Framework and can be run as server process.

Role of Database tampering detection management system in details are listed as follows:

1. to provide RESTfull services to clients for managing, executing and viewing results of the database tampering detection process - fulfils functional requirement FUNREQ.6 and FUNREQ.7;

2. to perform database tampering detection process steps - fulfils security requirement SECREQ.9, SECREQ.10, SECREQ.11, SECREQ.12 and functional requirements FUNREQ.6-8.

3.5.1 Working Principles of a Database Tampering Detection Management Software

A Database Tampering Detection Software is developed so it can be run on a system as an independent process. It connects to shared "H2" database file where cryptographic link meta-info is stored by the "Syslog-ng" module program and saves the database tampering detection process's results into the same database file. To perform the database tampering detection it can connect to the Database Management System where real data is stored. In Database Management System, there is located database procedures, that were developed during the research part of this thesis. Those procedures include functions to recalculate the database tables cryptographic link and retrieve resulting hash to management software. After the software receives the results, it compares the results with the information in the own database and stores validation result.

In the solutions proof of concept we implemented three different database tampering detection process flows. The first flow helps to validate the whole database table from the first to the last record based on the initial hash given by the system. The second flow helps to validate the section of the database table based on the initial hash, particular start and end row information. The third flow helps to give feedback about a possible tampered region in the database. Those flows are described in section 3.5.4. and on figure (See figure 14) and as follows:

1. When the system starts, it connects to the Audit Application System "H2" database, where cryptographic link information is collected;

2. When the system gets a request from the API or system scheduler it asks the corresponding cryptographic hash link info from the "H2" database to perform database tapering detection.

3. The system queries the Database Management Systems stored procedure for the cryptographic link hash regenerated by a specially developed database module.

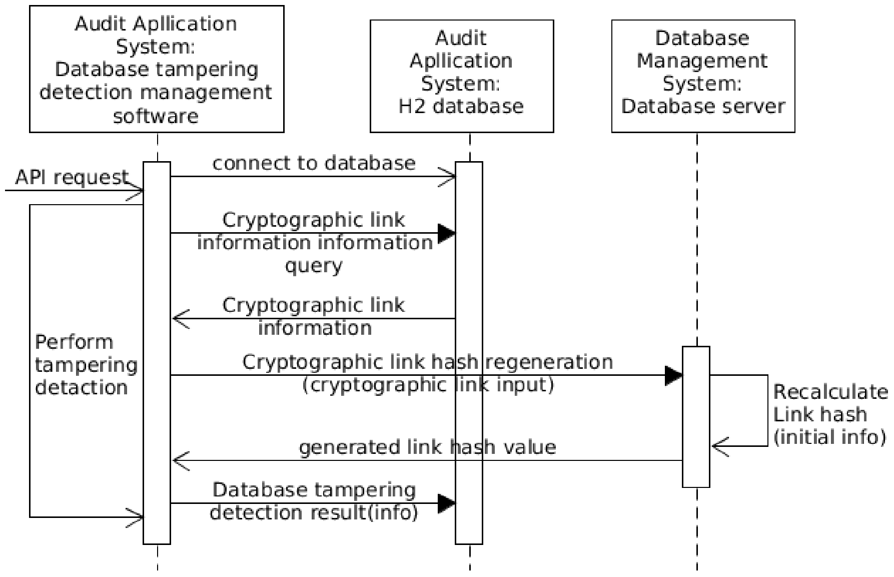

Audit Apllication System: Database tampering detection management software
Audit Apllication System: H2 database
Database Management System: Database server
API request
connect to database
Cryptographic link information information query
Cryptographic link information
Perform tampering detaction
Cryptographic link hash regeneration (cryptographic link input)
Recalculate Link hash (initial info)
generated link hash value
Database tampering detection result(info)

Figure 14: Detailed flow of the work of the Database Tampering Detection Management Software

4. The system performs a comparison of hashes stored in the "H2" database and retrieved from the Database Management System.
5. The system stores database tampering detection results into the "H2" database.

During the research of this thesis we did not developed a special user interface for the Database Tampering Detection Software, but the system can be controlled using REST API Framework "Swagger" [25] witch provides an automatically generated basic user interface to make REST requests that executes data tampering detection actions or presents tampering detection results from the database.

3.5.2 The Audit Application System Database Structure

The Audit Application System has one central database. We use the "H2" database for implementing data store needs. Firstly the database is shared between the "Syslog-ng" Java module program (see 3.4.2) for storing received cryptographic link information and has its own schema **auditdata**. Secondly the "H2" database is used by the database tampering detection management software with its own schema **audit**. Graphical representation of Audit Application System database structure is presented on figrue 15.

The database table **MAIN_DATA** consists of the received cryptographic link information. This information is parsed from the Syslog message discussed in section 3.4.2.

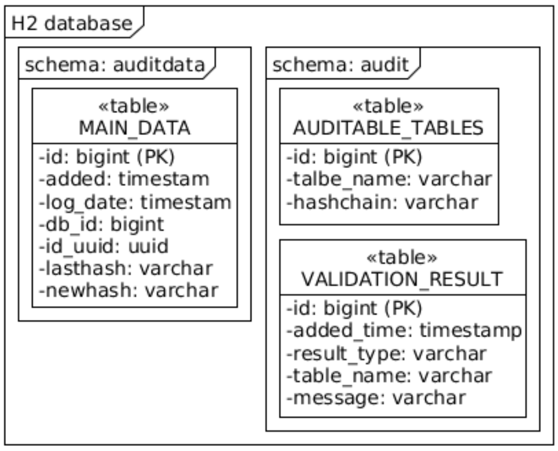


Figure 15: Database structure of Audit Application System

The "Syslog-ng" Java module program writes the data and the Database Tampering Detection Management Software reads from this table. For all auditable tables in the Database Management System a separate table must be created.

The database table **AUDITABLE_TABLES** contains info about auditable tables and the initial hashes of those tables. This table is used by Database Tampering Detection Management Software if the performing tampering detection of full table.

The database table **VALIDATION_RESULT** contains the results of the database tampering detection process. After every validation the Database Tampering Detection Management Software saves new validation report to this table.

3.5.3 Database Tampering Detection Management Software API Description

As described in section 3.2.1 we also use RESTfull API to provide ability for controlling application. API consists of one REST controller with the following endpoints:

- **/result/listByTableName** - HTTP GET request to get the list of the data tampering detection results by table name.
- **/validation/full** - HTTP GET request to execute the database table full tampering detection by table name.
- **/validation/partial** - HTTP GET request to execute the database table partial tampering detection by table name.
- **/validation/errorDetection** - HTTP GET request to execute the database table partial tampering detection with error detetion.

3.5.4 Database Tampering Detection Process

In this section we will discuss the database tampering detection more precisely to provide an explanation for security requirements SECREQ.9, SECREQ.10, SECREQ.11 and SECREQ.12 described in section 2.4.1. We provide three different database tampering detection flows, that are also implemented in the solutions proof of concept. With this section we will answer to research sub question RSQ 2.3.

From the database management perspective there can be different objectives and strategies to detect the possible tampering of secured data. All research work and discussions until now handle preparations to be able to detect possible database tamper. Nevertheless an important feature of this solution should be an understanding of how tampered data can be found using the provided solution.

To discover the fact of tamper or tampered elements from the database, we came up with the following database tampering detection process flows:

Tables full validation - This flow recalculates the database table cryptographic link hash based on the input hash from the first row until the last row. The input hash will be given to the database procedure and then the procedure delegates the input to the C language function what row by row takes data from the database table and calculates the hashes from data. At the end the C language function retrieves data to procedure what retrieves output to caller program.

input: The table initial hash - stored into the audit information system during table generation in DMBS.
output: The hash of the last row generated by the procedure.

Tables partial validation - This flow recalculates the database table cryptographic link hash based on the input hash the first table row and the last table row. The Input hash will be given to the database procedure and procedure delegates the input to the C language function that calculates hash link between two database rows based on input hash.

input: hash value
input 2: value of beginning record id
input 3: value of ending record id
output: hash of last record generated by procedure

Tables partial validation with tampered region detection - This flow recalculates the database table cryptographic link hash based on the input hash, the first table row and the last table row. It also validates at the same time, if the recalculated hash corresponds to the hash in the database tables corresponding column. If the hash does not match to hash on this particular record it retrieves the calculated hash and particular row information or nothing if the link is correct.

```
  input: hash value
input 2: value of beginning record id
input 3: value of ending record id
input 2: value of beginning record id
input 3: value of ending record id
 output: hash of last record generated by procedure
```

3.6 External Third Party Trustee

As we specified in section 3.1, an important component of our solution is the integration of an external third party trustee. During the research part of this thesis project we did not implement this counterpart, but during the development of the Information Application System we integrated a connection to the certification authority. For implementing security requirements SECREQ.1 to SECREQ.4 there is practically only one solution in Estonia to use third party certificate authority to generate digital signatures. In section 3.2.3 we described the implementation of the digital signature mechanism. To implement a digital signature, what would be valid according to the act described beforehand we need to use services from SK ID Solutions AS (previously named AS Sertifitseerimiskeskus). This company is a state accredited certificate and time stamp service provider. [21].

In a more practical manner, the digital signature creation and validation software library "Digidoc4j" that was discussed in section 3.2.3 is configured to communicate with services provided by SK ID Solutions AS. By conjunction with the security hardware controlled by the above mentioned library digital signature and time stamps are provided by the certification authority. For testing the proof of concept of this solution of this thesis project we received QSCD classified signature token "SafeNet 7300" from SK ID Solutions AS and a test certificate provided specially for our project to sign data with the test digital signatures.

It is also important to note, that every digital signature creation in production environment cost some money. This consumption must be planed into budget.

3.7 Summary

In this section we answered the research question RQ2 and sub-questions. Trough implementation steps and discussions of architectural components we provided the implementation of security and functional requirements. We gave background to technical design decisions and described in detail how the solution components interact with each other or what artefacts they provide. During the research of this section we developed a proof of concept solution to showcase and prove theories discussed and to fulfil the security objectives declared in section 2.

4 Proof of Concept Validation

The following section will provide a discussion and results of the validation of the solution against requirements produced in section 2. We will answer research question RQ3 and its sub questions. We will start our validation research with decoding the term "validation" and continue with a step-by-step process to ensure the solution implements requirements stated in the analyse phase.

4.1 Method

In this section we will answer research sub question RSQ 3.1. Validation is "The process of evaluating software at the end of the software development process to ensure compliance with software requirements" [50].

In this thesis we only focus on validating requirements and not the verification of each software component developed during the research. This is due to fact that we did not specifi any software specific requirements to follow during the development phase. But nonetheless during the development phase we applied several commonly known techniques and best practices, for example a code static analyse and inspections provided by software development environment tools.

For performing a solution validation, several operations can be taken including code, log files, data or database investigation, solution performance or behavioural investigation etc. A detailed list of used operations will be listed in the validation scenarios discussed in the following sections. The validation protocol steps for this research are the following:

1. For each requirement specified in section 2.4.1 and 2.4.2 acceptance criteria will be specified;
2. Validation scenarios will be specified that consist of several validation steps;
3. Validation scenarios will be conducted;
4. Validation scenario results will be presented.

On diagram 16 we present a set of artefacts and connections between them to perform a validation and make conclusions. Based on requirements and the solution we developed acceptance criterias for every requirement. Also we developed validation scenarios that consist of several validation steps that use input data and correspond to the acceptance criteria via expected output. If the scenario's steps do not have particular acceptance criterias, but it is needed for the next steps, we only define the expected output. Each scenario step produces an output and a scenario step result can be concluded from it and from the validation criteria. Scenarios results confirm that the requirements have been fulfilled.

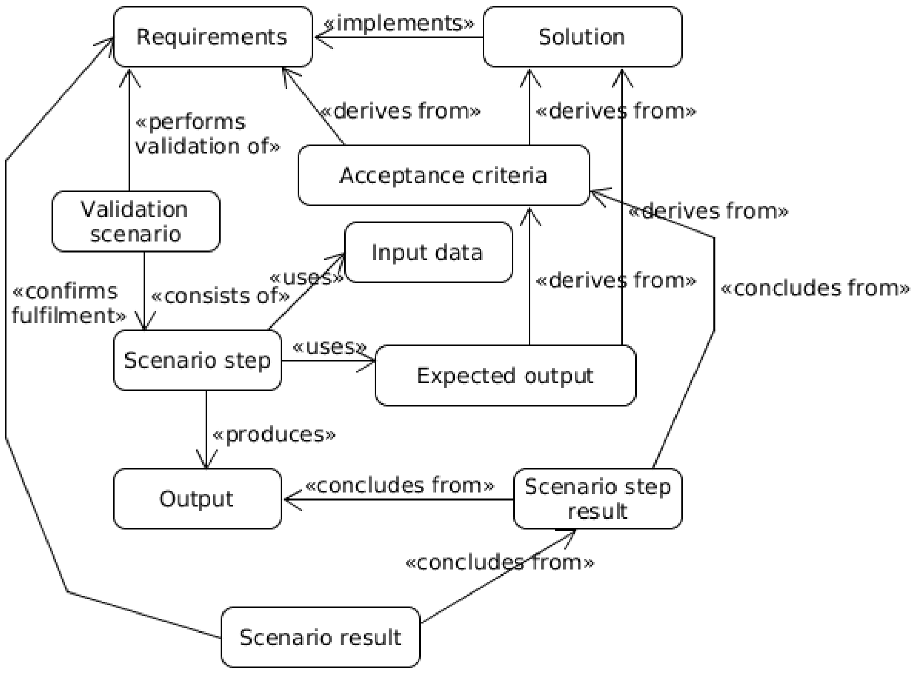


Figure 16: Validation artefacts and connections with requirements

The following sections will provide a detailed discussion about the validation steps listed above.

4.2 Defining Acceptance Criteria

For every requirement, an acceptance criteria is developed. If an acceptance criteria is conditionally me we can state that requirement is fulfilled. The acceptance criterias will have label of AC-SEC.NUMBER, where SEC stands for security requirement and the number corresponding requirement number. Acceptance criteria for security requirements are the following:

AC-SEC.1	A digital signature container retrieved from API data output operation has a digital signature container that can be opened by the Estonian digital signature software tool "digidoc-tool".
AC-SEC.2	A digital signature that accompanies the digital signature container, is in format that is presented with "BES/time-stamp" by Estonian digital signature software program "digidoc-tool".

AC-SEC.3	A digital signature must be based on public key cryptography. - with a digital signature container, consisting of certificates recognised by software "DigiDoc3 client" as valid certificates.
AC-SEC.4	A digital signature that accompanies with the digital signature container consists of a Signers Certificate that the software "Digidoc3 Client" presents with a Public key with at least RSA (1536).
AC-SEC.5	By investigating the source code, that is used to run the solution, there is a construction that ensures, that each following database record consist of the information that is cryptographically interpreted from the database record data and previous recor cryptographic information.
AC-SEC.6	By investigating the source code, that is used to run the solution, there is a construction that during the cryptographic linking of the database records for generating a hash digest to each following database record, the last database records hash is used alongside with the current databases record data to form a current record hash digest.
AC-SEC.7	By investigating the source code, that is used to run the solution, there is a construction that uses the SHA256 hash function to generate hash value from the databases record data.
AC-SEC.8	By investigating the Database Management Systems main data table record field "hashchain", the value is at least the length of 256 bits (it is equivalent to 32 bytes, or 64 bytes in an hexadecimal string format).
AC-SEC.9	By performing a database main data table full tampering detection with the solutions database tampering detection application, the Database Management Systems system log file a contains log row about performing this action and the tampering detection result is presented as an activity output.
AC-SEC.10	By performing the database main data table partial tampering detection with the solutions database tampering detection application, the Database Management Systems system log file contains a log row about performing this action and the tampering detection result is presented as an activity output.
AC-SEC.11	By performing the databases main data table partial tampering detection with error reporting with the solution databases tampering detection application, the Database Management System system log file contains log row about performing this action and the tampering detection result is presented as an activity output.

AC-SEC.12	By investigating the database records of the Audit Application System, there are records that indicate a direct connection with main data records in the Database Management System.

Acceptance criterias for functional requirements will have the label of AC-FUN.NUMBER, where FUN stands for functional requirement and the number corresponds to the requirement number. Acceptance criteria for functional requirements are the following:

AC-FUN.1	After executing API operations, a response is available.
AC-FUN.2	After executing API operations for main data input procedures, data is stored in the Database Management System.
AC-FUN.3	After executing API operations for data output procedures, system outputs requested the main data.
AC-FUN.4	By executing API operations for the output main data, the signature validation result is presented as part of the output data.
AC-FUN.5	By connecting to the Database Management System database with "PostgreSQL" client software, a table with the name "main_data" is found.
AC-FUN.6	By calling the system module API operations, it is possible to initiate a database table's full detection, partial detection and partial detection with tampered record detection execution.
AC-FUN.7	By calling system module API operations it returns the database tampering detection results in JSON formatted data.
AC-FUN.8	By calling solution API operations and executing database queries in the "H2" database table "results", tampering detection results are presented.

4.3 Scenarios For Validation

In this section we describe a set of scenarios to validate that the requirements are fulfilled. Stakeholder can use the step-by-step flow to ensure that requirements are fulfilled. In each scenario section we will provide brief background information to understand the scenario's context. Detailed scenarios with sample input data, expected output what is derived from the acceptance criteria and implementation specifics are provided in appendix II. Validation scenarios. For the scenario to be accomplished with a positive result, all the scenario steps must succeed.

SCEN1 - Scenario for validating data insertion and cryptographic hash chain creation

This scenario is designed and developed to validate requirements connected to the main data input operations and main data digital signing in the Application Information System The creation of a cryptographic link of main data records in the Database Application System and deposition of cryptographic link meta data to the Audit Application System. In this scenario we use the HTTP client tool, "PostgreSQL" and "H2" database client tool and the Estonian digital signature software "DigiDoc3 Client".

The scenario validates the following requirements: SECREQ.8, SECREQ.12, FUNREQ.1, FUNREQ.2, FUNREQ.5.

SCEN2 - Scenario for validating data output to client, file content and digital signature features
This scenario is designed and developed to validate requirements connected to the main data output operations from the Information Application System and the main data protecting digital signature's verification. In this scenario we use the HTTP client tool and the Estonian digital signature software "digidoc-tool" and "DigiDoc3 Client".

The scenario validates the following requirements: SECREQ.1, SECREQ.2, SECREQ.3, SECREQ.4, FUNREQ.3, FUNREQ.4.

SCEN3 - Scenario for validating data tampering detection
This scenario is designed and developed to validate requirements connected to data tampering detection execution in the Audit Application System and the data tampering detection process by the Database Tampering Detection Management Software and the Database Management Software. In this Scenario we use the HTTP client tool and "PostgreSQL" database client tool.

The scenario validates the following requirements: SECREQ.9, SECREQ.10, SECREQ.11, FUNREQ.6, FUNREQ.7, FUNREQ.8.

SCEN4 - Scenario for validating cryptographic link generation
This scenario is designed and developed to validate requirements connected to the cryptographic link generation in the Database Management Software. Scenario is based on the source code inspection and does not consist of any steps that is doing execution of the software implemented in this thesis project, because it will not cover any functional requirements.

Scenario validates following requirements: SECREQ.5, SECREQ.6, SECREQ.7.

4.4 Performance Characteristics of Data Tampering Detection

In this section we will answer research sub question RSQ 3.2. At the end of the validation process we performed several tests to get the performance characteristics of database tampering detection process. We particularly measured the time of performing tampering detection on records for ten times in test. Regardless of the fact, that in this thesis we offered three database tampering processes, the tests we only performed on the "Table partial validation" process. Both of the other process are similar to the one algorithm. Only "Table partial validation with tamper region detection" process has some additional operations inside the algorithm, that could make it slightly slower. For a test environment we used a Virtualized Linux server running "PostgreSQL" version 9.6.2. This virtual server has a 64 bit processor Intel Core i7-2640M at 2.80GHz with 4 cores and 4000MB of RAM. In the databases main table we had 1 million records cryptographically linked. We performed tests using database function "getPartialTableHashFunctionPublic" with SQL query:

```
select * from getPartialTableHashFunctionPublic(?,?,?,?);

with parameters:
1st = number 1
2nd = id of record to perform tampering detection
      (for example 1000, 10000, 100000, 1000000)
3rd = text of initial hash from cryptographic link
      generation shared object library
4th = text 'main_data'
```

In table 1 we provide the average test time in seconds. On figure 17 graphical representation of the test results are shown.

Table 1: Average time consumption for performing tampering detection

Number of records in process	**Time for process in seconds**
1000	0.011
10000	0.116
50000	2
100000	8
200000	31
300000	71
400000	128
500000	197
600000	283
700000	378
800000	487
900000	610
1000000	754

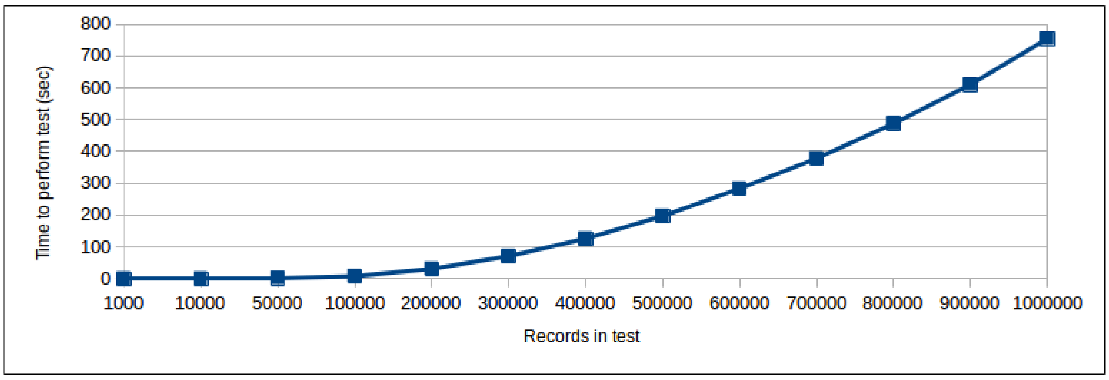


Figure 17: Performance test results average time diagram

According to the given data we can state that the more records are involved in one detection process the more it time takes to finish the detection process. For a large number of records tamper detection can last for hours fro full table detection, so we conclude that full database tampering process can be inefficient. But this can be avoided if more powerful processor is used. It is also possible to divide database table to regions of interest and run the detection based on the mentioned regions in parallel.

4.5 Validation Results

In this section we will answer research sub-question RSQ 3.3. During the validation process we improved the software developed during the thesis project. Validation scenarios and the overall validation process helped to improve the quality of developed software. Also during validation, we ensured that the developed software meets all the security and functional requirements stated in section 2. We conclude, that after thew validation process is performed the solution meets above-mentioned requirements and the solution can be used in real life.

4.6 Summary

In this section we discussed the validation of the solution provided by this thesis research project and answered the research question RQ3 and sub-questions. We provided method for ensuring that the requirements specified in section 2 are correctly implemented. The method we provided acceptance criteria for every requirement and developed validation scenarios to cover the validation of requirements. We also provided performance characteristics of time consumption for the data tampering detection process. At the end of this section we provided the results of the validation process.

5 Application Guidlines

In this section we provide a discussion and instructions applying the solution what was developed in this thesis research. In appendix III. Solution application instruction details we provide details for configuring critical parts of our solution, building instructions to the developed software and running instructions to solutions subsystems. A sample configuration for the applications are included in the source code of this solution and not added in the appendixes.

5.1 Background Discussion

Every proof of concept without additional improvements is not possible to apply to the provided solution. Our solution covers the digital signature concept, database hash link generation, cryptographic link meta data deposition and database tampering detection. Subjects that we did not cover were database internal structure and data object structure. In section 3.3.1 we discussed how we store data in the Database Management System, but it is also important to have versioning of data sets, that is not covered in this thesis. To apply improvements must be made to the main data table to allow full versioning support for the database objects. According to database main table changes, the database cryptographic hash chain source code must be changed and compiled.

Also some organisational changes are needed to apply this solution. We propose, that Audit Application System of this solution should be deployed in different administrative and network zones than the Database Management System. There should be different administrators and access roles for administering the Database Management System and the Audit Application System components. If same person has access to the Audit Application System and the Database Management System, there is risk that the cryptographic link meta data saved into Audit Application System by administrator is manipulated.

5.2 Necessary Software Components for Solution

In the following section we list the software needed to install the solution. We do not provide a detailed step-by-step command list for installing the whole solution, because this can wary based on the operating system used. Build instructions for developed software is presented in appendix III. Solution application instruction details.

Information Application System

- Linux operating system. In this research project we used 64bit Ubuntu 16.04.2 LTS

- "Java 8 SE Runtime Environment". In the research project we used version 1.8.0_131-b11
- pkcs11 device integration client. In this research project we used "SafenetAuthenticationClient-core" version 9.1.7.
- Main application (ais), that developed during this research project.

Database Management System

- Linux operating system. In this research project we used 64bit Ubuntu 16.04.2 LTS
- "PostgreSQL" version 9.5 or later. In this research project we used "PostgreSQL" version 9.6.2.
- "Syslog-ng2" version 3.5 or later. In this research project we used "Syslog-ng" version 3.5.6.
- Database cryptographic linking and data tampering detection shared object library "hashchain.so", that was developed during this research project.

Audit Application System

- Linux operating system. In this research project we used 64bit Ubuntu 16.10
- "Syslog-ng" version 3.9 or later with mod-java and mod-java-common compiled. In research project we used "Syslog-ng" version 3.9.1.
- "Java 8 SE Runtime Environment". In this research project we used version 1.8.0_131-b11.
- Database tampering detection management software "audit", that was developed during this research project.
- "Syslog-ng" Java module "H2Writer", that was developed during this research project.

6 Conclusion

In this thesis project we analysed, developed and validated solution to provide proof of concept how to implement selected ISKE highest level integrity requirements. It is crucial that mentioned security requirements understood in early design and the development phase. Later modifications of the system and software almost equals developing of new system. This research work will give an understanding about the background of the selected ISKE integrity measures that have effect to system in the early design and software development phase. Research provide a validation method to ensure how elicited requirements are fulfilled. During the research project of this thesis we developed a software that after improvements can be applied as an info technology system to securely store data that has ISKE integrity subclass "T3".

6.1 Answers to Research Questions

Using experience and data collected during this research project we can conclude the following:

RQ1. - How should the proof of concept be designed to obtain security objectives of data with highest integrity need demanded by ISKE? - During every development phase of the information system, it is required to have a detailed analyse of following aspects: organisational context and assets, security risks, risk treatment decision, security and functional requirements and what are the possible limitations of the requirements. In our thesis project, we created a fictional state registry that describes a real world example as closely as possible. We researched over the listed aspects and collected set of requirements for this fictional system based on the selected ISKE highest level integrity security measures. During this process we used the Information System Security Risk Management (ISSRM) methodology in conjunction with ISKE Reference Guide.

RQ2. - How should the proof of concept and the system implemented to obtain security objectives of data with highest integrity need demanded by ISKE? - During the research we implemented a proof of concept solution to demonstrate how to achieve objectives stated in research question RQ1. We divided the solution into three subsystems that have dependencies and all together fulfil the security requirements and help to achieve the security objectives. There is a sub system that deals with user requests for data input and output operations and is responsible for data signing with a digital signature. Secondly, there is a database system that stores data given by the first subsystem. Thirdly, there is a subsystem that manages data tampering detection and collects deposited cryptographic link data. All three systems can be given under different administrative zones to ensure administration separation. In order to still have flexibility in the database level we provided a database table structure using JSON and

"PostgreSQL" Database Management System capable of allowing dynamic organisation of stored data objects. This approach is called NoSQL and is a representation of a not-traditional database strategy. To ensure long term integrity and legal aspects we provided a solution to encapsulate data into digital signature containers specified with EU regulation on electronic identification and trust services for electronic transactions in the internal market.

RQ3. - How should the solution validation be performed to ensure the fulfilment of security requirements? - We provided a validation method to ensure that the security and functional requirements are fulfilled by the implemented solution. We developed acceptance criteria for each requirement and validation scenarios including several steps with input data and expected output. We performed the validation steps and during this process improved the softwares quality. We also researched performance characteristics of the database tempering detection process. Overall we concluded that the implemented solution fulfils requirements decalred in the research project.

6.2 Limitations

By applying the provided solution it is needed take into consideration that limitations and effects come with the declared security requirements. Using this solution drastically decreases the flexibility of the database structure and the ability to make changes on table structure. As ISKE introduces term chronologically ordered hash link of database records, it means that any intended changes to the database records are not possible. There can be only insert-only database, where changes in the database objects have to be organised logically and therefore a versioning of records is needed. This aspect means, that database can grow very large in count of records depending on business processes and the fundamental life cycle of the data object.

In the database tampering detection performance test we concluded, that the full database tampering detection process can be inefficient. One solution to overcome this is to paralelize tampering detection by dividing a larger area of interest into smaller regions and running detection on different threads.

6.3 Concluding Remarks

Work on this thesis project was started in September of 2015. By the time of publication this work, there is already a state information system that is partly implements using hints and suggestions provided by the author of this thesis. By providing analysis about the subject domain, theoretical solution design and implementation of proof of concept, we hope it is possible to help institutions and info technology developers to obtain a system that meets security requirements required by ISKE. By publishing this thesis and potential solution as proof of concept, it can enlarge ISKEs application participants awareness in the subject domain concepts.

6.4 Future Work

According to the understating of the author, proof of concept is 20% of product that is created with 80% of time or vice versa. During research we get different new ideas and questions that should be considered as follow-up research and discussion. Those aspects can be considered as research questions or point of improvement for future work:

- Are there other mechanisms that can be used to determine unintentional changes in databases for example blockchain technology?
- Is there a possibility to improve ISKE to allow usage of above mentioned possible technologies?
- How the provided database tampering detection process can be optimised to increase the speed of tampering detection of a very large quantity of records?
- What is the possible process of database recovery when database tampering is detected?
- Improve the Database Tampering Detection Management Software to be more flexible for profile scenarios based detection;
- Improve the Database Tampering Detection Management Software to automatically perform profiles based detection including notification about detected violations.

References

[1] Estonian Parliament. Infosüsteemide turvameetmete süsteem, 2008. [Online]. Available: `https://www.riigiteataja.ee/akt/13125331`. [Accessed: 01-Dec-2016].

[2] Information System Authority Estonia. Three-level it baseline security system iske, 2012. [Online]. Available: `https://www.ria.ee/en/iske-introduction.html`. [Accessed: 01-Dec-2016].

[3] Information System Authority Estonia. Infosüsteemide kolmeastmeline etalonturbe süsteem iske rakendusjuhend, 2016. [Online]. Available: `https://iske.ria.ee/8_01/?action=AttachFile&do=get&target=ISKE_rakendusjuhend_8.00.pdf`. [Accessed: 01-Dec-2016].

[4] Information System Authority Estonia. Infosüsteemide kolmeastmeline etalonturbe süsteem iske kataloogid, 2016. [Online]. Available: `https://iske.ria.ee/8_01/ISKE_kataloogid`. [Accessed: 01-Dec-2016].

[5] Information System Authority Estonia. Infosüsteemide kolmeastmeline etalonturbe süsteem iske ohud, 2016. [Online]. Available: `https://iske.ria.ee/8_01/ISKE_ohtude_kataloog`. [Accessed: 01-Dec-2016].

[6] É. Dubois, P. Heymans, N. Mayer, and R. Matulevičius. A systematic approach to define the domain of information system security risk management. page pp. 289306, Springer, 2010.

[7] R. Matulevičius, N. Mayer, H. Mouratidis, P. Heymans, and E. Dubois. Syntactic and semantic extensions to secure tropos to support security risk management. pages 18(6):816–844, 2012.

[8] G. Sindre and A. L. Opdahl. Eliciting security requirements with misuse cases. page 10(1):34–44, 2005.

[9] R. Matulevicius, N. Mayer, and P. Heymans. Alignment of misuse cases with security risk management. In *Availability, Reliability and Security*, 2008.

[10] Information System Authority Estonia. G 4.13 salvestatud andmete hävimine. [Online]. Available: `https://iske.ria.ee/8_01/ISKE_ohtude_kataloog/G4/G_4.13`. [Accessed: 01-Dec-2016].

[11] Information System Authority Estonia. G 4.28 andmebaasi andmekadu, 2016. [Online]. Available: `https://iske.ria.ee/8_01/ISKE_ohtude_kataloog/G4/G_4.28`. [Accessed: 01-Dec-2016].

[12] Information System Authority Estonia. G 4.30 andmebaasi tervikluse ja vastavuse kadu, 2016. [Online]. Available: https://iske.ria.ee/8_01/ISKE_ohtude_kataloog/G4/G_4.30. [Accessed: 01-Dec-2016].

[13] Information System Authority Estonia. G 5.64 andmete või tarkvara manipuleerimine andmebaasisüsteemides, 2016. [Online]. Available: https://iske.ria.ee/8_01/ISKE_ohtude_kataloog/G5/G_5.64. [Accessed: 01-Dec-2016].

[14] D. Firesmith. Specifying good requirements. 2:77–87, 2003.

[15] Information System Authority Estonia. Ht.10 andmebaasi kannete krüptoahelدamine, 2016. [Online]. Available: https://iske.ria.ee/8_01/ISKE_kataloogid/8_Kataloog_H/HT/HT.10. [Accessed: 01-Dec-2016].

[16] Information System Authority Estonia. Ht.34 digiallkirja kasutamine, 2016. [Online]. Available: https://iske.ria.ee/8_01/ISKE_kataloogid/8_Kataloog_H/HT/HT.34. [Accessed: 01-Dec-2016].

[17] Information System Authority Estonia. Ht.52 lisanõuded krüptovahenditele, 2016. [Online]. Available: https://iske.ria.ee/8_01/ISKE_kataloogid/8_Kataloog_H/HT/HT.52. [Accessed: 01-Dec-2016].

[18] Estonian Parliament. Electronic identification and trust services for electronic transactions act, 2016. [Online]. Available: https://www.riigiteataja.ee/akt/125102016001. [Accessed: 01-Dec-2016].

[19] M. J. Malmgren. An infrastructure for database tempering detection and forensic analysis. Bathelor thesis, The Univeristy of Arizona, 2007.

[20] R. Snodgrass, S. S. Yao, and C. Collberg. Tamper detection in audit logs. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, VLDB '04, pages 504–515. VLDB Endowment, 2004.

[21] AS Sertifitseerimiskeskus. Services, 2017. [Online]. Available: https://www.sk.ee/en/services/. [Accessed: 01-Dec-2016].

[22] F. Gutierrez. *Introducing Spring Framework*. 2014.

[23] T. Fredrich. *RESTful Service Best Practices*, 2012.

[24] R. T. Fielding. Architectural styles and the design of network-based software architectures. Technical report, UNIVERSITY OF CALIFORNIA, University of California, Irvine, 2000.

[25] Swagger development community. *Documenting an Existing API with Swagger*, 2017. http://swaggerhub.com/wp-content/uploads/2017/02/Documenting-An-Existing-API-with-Swagger-2.pdf.

[26] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta. Comparison of json and xml data interchange formats: A case study, department of computer science montana state university – bozeman. 2009.

[27] Z. U. Haq, G. F. Khan, and T. Hussain. A comprehensive analysis of xml and json web technologies. 2015.

[28] J. Popovic. Storing json in sql server, 2015. [Online]. Available: https://blogs.msdn.microsoft.com/sqlserverstorageengine/2015/11/23/storing-json-in-sql-server/. [Accessed: 01-Dec-2016].

[29] Oracle Corporation. *Schemaless Application Development with Oracle Database 12c*, 2015.

[30] C. Bienko, M. Greenstein, S. E. Holt, R. T. Phillips, and IBM International Technical Support Organization. *IBM Cloudant: Database as a Service Fundamentals*. 2015. http://www.ibm.com/developerworks/data/library/techarticle/dm-1306nosqlforjson1/.

[31] C. Chasseur, Y. Li, and J. M. Patel. Enabling json document stores in relational systems. 2013.

[32] The PostgreSQL Global Development Group. Json types, 2016. [Online]. Available: https://www.postgresql.org/docs/current/static/datatype-json.html. [Accessed: 01-Dec-2016].

[33] AS Sertifitseerimiskeskus. The estonian id card and digital signature concept, 2003. [Online]. Available: http://www.id.ee/public/The_Estonian_ID_Card_and_Digital_Signature_Concept.pdf. [Accessed: 01-Dec-2016].

[34] Estonian Information System Authority. Digidoc4j api description, 2016. [Online]. Available: http://open-eid.github.io/digidoc4j/. [Accessed: 01-Dec-2016].

[35] Information System Authority Estonia. Digital signature formats, 2016. [Online]. Available: http://www.id.ee/index.php?id=36108. [Accessed: 01-Dec-2016].

[36] The PostgreSQL Global Development Group. Postgresql, 2016. [Online]. Available: https://www.postgresql.org/. [Accessed: 01-Dec-2016].

[37] F. Gessert, W. Wingerath, S. Friedrich, and N. Ritter. Nosql databases: a survey and decision guidance. Research paper, University of Hamburg, Germany, 2016.

[38] C. Strauch. Nosql databases. resreport, Stuttgart Media University, 2011.

[39] G. Litt, S. Thompson, and J. Whittaker. Improving performance of schemaless document storage in postgresql using bson. Research paper, Yale University, 2013.

[40] M. Nenciarini. Jsonb type performance in postgresql 9.4, 2015. [Online]. Available: http://blog.2ndquadrant.com/jsonb-type-performance-postgresql-9-4/. [Accessed: 01-Dec-2016].

[41] The PostgreSQL Global Development Group. C-language functions, 2016. [Online]. Available: https://www.postgresql.org/docs/current/static/xfunc-c.html. [Accessed: 01-Dec-2016].

[42] K. Kapp. Kõrge terviklusega andmeid talletava andmebaasilahenduse prototüüp. Associates thesis, Tallinn Polytechnic Shool, 2016.

[43] D. Eastlake. Us secure hash algorithms (sha and hmac-sha), 2006. [Online]. Available: https://tools.ietf.org/html/rfc4634. [Accessed: 01-Dec-2016].

[44] The syslog protocol, 2009. https://tools.ietf.org/html/rfc5424.

[45] Adiscon GmbH. Rsyslog, 2016. [Online]. Available: http://www.rsyslog.com. [Accessed: 01-Dec-2016].

[46] Balabit SA. syslog-ng - open source log management solution, 2017. [Online]. Available: https://syslog-ng.org/. [Accessed: 01-Feb-2017].

[47] R. Gerhards. omprog: Program integration output module, 2016. [Online]. Available: http://www.rsyslog.com/doc/v8-stable/configuration/modules/omprog.html. [Accessed: 01-Dec-2016].

[48] R. Gerhards. omlibdbi: Generic database output module, 2016. [Online]. Available: http://www.rsyslog.com/doc/v8-stable/configuration/modules/omlibdbi.html. [Accessed: 01-Dec-2016].

[49] Balabit SA. Reliable log transfer protocol^{TM}, 2017. [Online]. Available: `https://www.balabit.com/documents/syslog-ng-pe-5.0-guides/en/syslog-ng-pe-guide-admin/html/concepts-rltp.html`. [Accessed: 01-Feb-2017].

[50] B. W. Boehm. Verifying and validating software requirements and design specifications. *IEEE Softw.*, 1(1):75–88, January 1984.

Appendix

I. The cryptographic hash chain generation source code

```
#include <string.h>
#include <syslog.h>
#include <openssl/sha.h>
#include <postgres.h>
#include <executor/spi.h>
#include <utils/rel.h>
#include <utils/builtins.h>
#include "hashchain.h"

static const char hexdigits[16] = "0123456789abcdef";
static const char hashchainFieldName[] = "hashchain";

/**
 * loads & returns hash of previous row in the relation
 * than once per table per server startup.
 * @param relationName name of relation/table
 * @returns hashchain value of last row of the relation
 * @note returned pointer points to statically allocated memory
 */
const char* loadHash(const char relationName[NAMEDATALEN])
{
  static const char prevHashQueryTemplate[] = "SELECT hashchain FROM
      \"%s\" ORDER BY id DESC LIMIT 1";
  static const char initialHash[] = "160488245
      c65a2f36685f107e1f499caac7f37dc6afb65a4ed9cc9f300b30f05";
  static char retHash[SHA256_DIGEST_LENGTH * 2 + 1] = "";
  static char query[NAMEDATALEN + sizeof(prevHashQueryTemplate)] = ""
      ;

  openlog("hash_chain", LOG_NDELAY, LOG_SYSLOG);
  syslog(LOG_DEBUG, "Trying to load hash from table \"%s\"",
      relationName);
  snprintf(query, sizeof(query), prevHashQueryTemplate, relationName)
      ;
  if(SPI_connect() == SPI_ERROR_CONNECT) {
    syslog(LOG_ERR, "Failed to load hash from \"%s\": SPI_connect()
        == SPI_ERROR_CONNECT", relationName);
    return NULL;
  }
  int r = SPI_execute(query, true, 0);
  if(r != SPI_OK_SELECT) {
    syslog(LOG_ERR, "Failed to load hash from \"%s\": r !=
        SPI_OK_SELECT", relationName);
    SPI_finish(); // ignore SPI_ERROR_UNCONNECTED error
    return NULL;
  }
  if(SPI_processed != 1) {
    SPI_finish(); // ignore SPI_ERROR_UNCONNECTED error
    syslog(LOG_ERR, "No rows returned from  \"%s\", using initial
       hash: %s", relationName, initialHash);
    return initialHash;
  }
  HeapTuple row = SPI_tuptable->vals[0];
  TupleDesc desc = SPI_tuptable->tupdesc;
  char* hash = SPI_getvalue(row, desc, 1);
  memcpy(retHash, hash, SHA256_DIGEST_LENGTH * 2 + 1);
  pfree(hash);
  SPI_finish();
  syslog(LOG_DEBUG, "Hash loaded from \"%s\": %s", relationName,
     retHash);
  return retHash;
}

/**
 * @param rel relation (table)
 * @param row row to be inserted
 * @returns row with hashchain field filled
 * @note Postgre server takes care of freeing allocated memory
 * @note elog statements seem to have significant performance impact
 */
HeapTuple hashchain(const Relation rel, const HeapTuple row)
{
  const char* tableName = SPI_getrelname(rel); // table name
  int hashchainFieldNumber = SPI_fnumber(rel->rd_att,
     hashchainFieldName); // hashchain field index
  int idFieldNumber = SPI_fnumber(rel->rd_att, "id"); // id field
     index
  int idUuidFieldNumber = SPI_fnumber(rel->rd_att, "id_uuid"); //
     id_uuid index

  openlog("hash_chain", LOG_NDELAY, LOG_SYSLOG);

  if(hashchainFieldNumber <= 0) {
    elog(INFO, "hashchain triggered on table without hashchain field
       (%s)", tableName);
    return NULL; // not hashchain'ed, no-op
  }
  const char* prevHash = loadHash(tableName);
  if(prevHash == NULL) {
    elog(ERROR, "Couldn't get last hash (see system log)");
    return NULL;
  }
  /* initialize sha256 context */
  SHA256_CTX sha256;
  SHA256_Init(&sha256);
  SHA256_Update(&sha256, prevHash, SHA256_DIGEST_LENGTH * 2);

  /* add all fields to digest */
  int i;
  for(i = 1; i <= rel->rd_att->natts; i++)
    {
    if(i == hashchainFieldNumber) continue; // except hashchain field
    char* fieldValue = SPI_getvalue(row, rel->rd_att, i);
        if ((fieldValue != NULL && fieldValue[0] != '\0')) {
            SHA256_Update(&sha256, fieldValue, strlen(fieldValue));
        }
  }

  /* first 32 bytes of output will be binary representation of digest
      */
  unsigned char hash[SHA256_DIGEST_LENGTH * 2 + 1];
  SHA256_Final(hash, &sha256);

  /* convert binary representation to string */
  for(i = SHA256_DIGEST_LENGTH - 1; i >= 0; i--) {
    hash[i * 2 + 1] = hexdigits[hash[i] & 0x0f];
    hash[i * 2] = hexdigits[(hash[i] & 0xf0) >> 4];
  }
  hash[SHA256_DIGEST_LENGTH * 2] = '\0'; // string terminator

  /* send hash & metadata to log server */
  char* rowId = SPI_getvalue(row, rel->rd_att, idFieldNumber);
  char* rowUuid = SPI_getvalue(row, rel->rd_att, idUuidFieldNumber);
    syslog(LOG_NOTICE, "hashChainAudit:%s,%s,%s,%s,%s", tableName,
       rowId, rowUuid, prevHash, hash);

  /* create new tuple with hash field filled */
  int columns[1] = { hashchainFieldNumber };
  Datum values[1] = { CStringGetTextDatum(hash) };
  HeapTuple modifiedTuple = SPI_modifytuple(rel, row, 1, columns,
     values, NULL);

  if(SPI_result == SPI_ERROR_NOATTRIBUTE) {
    elog(ERROR, "SPI_result == SPI_ERROR_NOATTRIBUTE");
    return 0;
  }
  closelog();
  return modifiedTuple;
}
```

II. Validation scenarios

SCEN1 - Scenario for validating data insertion and cryptographic hash chain creation

Scenario validates following requirements: SECREQ.8, SECREQ.12, FUNREQ.1, FUNREQ.2, FUNREQ.5.

1. By using REST client or Swagger UI provided by solution make HTTP POST request with JSON data set to Information Application System API endpoint /postIdentity.

   Requirement: FUNREQ.1
   Input:

   ```
   curl -X POST --header 'Content-Type: application/json' \
       --header 'Accept: */*' \
       -d '{"firstName": "John", "lastName": "Smith", "age":
   25}' \
       'http://localhost:8080/postIdentity'
   ```

   Acceptance criteria: AC-FUN.1
   Expected output: JSON formated data containing elements idUuid, historyIdUuid, mainData.
   Output:

   ```
   { "idUuid": "6c7d30f6-269c-407e-a942-9855f7818386",
     "historyIdUuid": null,
     "mainData": "{\n  \"firstName\": \"John\",\n  \"lastName\":
   \"Smith\",\n  \"age\": 25\n}",
     "file": "UEsDBAoAAAgAADeTiU..." }
   ```

   Scenario step result: Successful

2. By using PostgreSQL database client tool make SQL select query to Database Management System database containing table named "main_data".

   Requirement: FUNREQ.5
   Input:

   ```
   SELECT id FROM main_data LIMIT 1;
   ```

   Acceptance criteria: AC-FUN.5
   Expected output: SQL result set contains one row with element "id" and data.

Output: "id" with data.
Scenario step result: Successful

3. By using PostgreSQL database client tool make SQL select query to Database Management System database containing table named "main_data". From SQL select result set, extract data from field named "file" and convert with BASE64 decode tool to file with file name "scenario1.asice".

   Requirement: no requirement
   Input:

   ```
   SELECT file FROM public.main_data
     WHERE id_uuid='6c7d30f6-269c-407e-a942-9855f7818386';
   ```

   Acceptance criteria: no criteria
   Expected output: file with name "scenario1.asice"
   Output: file with name "scenario1.asice"
   Scenario step result: Successful

4. Open file "scenario1.asice" with Estonian digital signature software "Digidoc3 client" and save file "data.json" to file system. Compare input data in step 1 with data in file "data.json".

   Requirement: FUNREQ.2
   Input:

   ```
   data posted in scenario SCEN1 step 1:
   {
     "firstName": "John",
     "lastName": "Smith",
     "age": 25
   }

   content of file "data.json" in digital signature container
   with name "scenario1.asice":
   {
     "firstName": "John",
     "lastName": "Smith",
     "age": 25
   }
   ```

   Acceptance criteria: AC-FUN.2
   Expected output: data posted in step 1 is equal to content of file "data.json"

Output: Both, main data of input operation matched with data contained in database.
Scenario step result: Successful

5. By using PostgreSQL database client tool make SQL select query to Database Management System database containing table named "main_data" to get result set with inserted main data.

   Requirement: FUNREQ.2
   Input:

   ```
   SELECT main_data FROM public.main_data
     WHERE id_uuid='6c7d30f6-269c-407e-a942-9855f7818386';

   data posted in scenario SCEN1 step 1:
   {
     "firstName": "John",
     "lastName": "Smith",
     "age": 25
   }
   ```

   Acceptance criteria: AC-FUN.2
   Expected output: SQL result set with field "main_data" and data posted in API input operation is equal.
   Output: Both, main data of input operation matched with data contained in database.
   Scenario step result: Successful

6. By using PostgreSQL database client tool make SQL select query to Database Management System database containing table named "main_data". From SQL select result set, extract data from field "hashchain" and count characters of data extracted for field "hashchain".

   Requirement: SEQREQ.8
   Input:

   ```
   SELECT hashchain FROM public.main_data
     WHERE id_uuid='6c7d30f6-269c-407e-a942-9855f7818386';
   ```

   Acceptance criteria: AC-SEQ.8
   Expected output: String of 64 characters.
   Output:

   ```
   4cf35543507a382d3694a1c5ffd236e5046cda91f29e902f74bdb14276335eec
   ```

   Scenario step result: Successful

7. By using PostgreSQL database client tool make SQL select query to Database Management System database containing table named "main_data". And by using H2 database client tool make SQL select query to Audit Application System H2 database containing table named "MAIN_DATA".

   Requirement: SECREQ.12
   Input:

```
Database Management System SQL query:
SELECT id, id_uuid, hashchain, file
    FROM public.main_data
    ORDER BY id DESC
    LIMIT 2;

H2 database SQL query:
SELECT id_uuid, db_id, last_hash, new_hash
    FROM main_data
    ORDER BY id DESC
    LIMIT 1;
```

   Acceptance criteria: AC-SEC.12
   Expected output: In result set of H2 database SQL query field "db_id" is equal to Database Management System SQL query result set first row field "id", H2 database SQL query result set field "id_uuid" is equal to Database Management System SQL query result set first row field "id_uuid", H2 database SQL query result set field "last_hash" is equal to Database Management System SQL query result set second row field "hahschain", H2 database SQL query result set field "new_hash" is equal to Database Management System SQL query result set first row field "hashchain".
   Output:

```
Result of H2 database SQL query:
id_uuid=6c7d30f6-269c-407e-a942-9855f7818386
db_id=1012
last_hash=
5d07171cbfc747786055517004b6ce24d20d90ec8d5a680af14a784e6ef05f6e
new_hash=
4cf35543507a382d3694a1c5ffd236e5046cda91f29e902f74bdb14276335eec

Result of Database Management System SQL query:
first row:
id=1012
id_uuid=6c7d30f6-269c-407e-a942-9855f7818386
```

```
hashchain=
4cf35543507a382d3694a1c5ffd236e5046cda91f29e902f74bdb14276335eec
file=UEsDBAoAAAgAADeTiUqKIflFHwAAAB........
second row:
id=1011
id_uuid=fa45192d-61a0-4257-8e47-d40779be6971
hashchain=
5d07171cbfc747786055517004b6ce24d20d90ec8d5a680af14a784e6ef05f6e
file=UEsDBAoAAAgAABWLiUqKIflFHwAAAB........
```

Scenario step result: Successful

SCEN2 - Scenario for validating data output to client, file content and digital signature features

1. By using REST client or Swagger UI provided by solution make HTTP POST request with JSON data set to Information Application System API endpoint /getIdentity.

   Requirement: FUNREQ.3, FUNREQ.4

   Input:

   ```
   curl -X GET --header 'Accept: application/json' \
       'http://localhost:8080/getidentity?identity_uuid=6c7d30f6-269c-
   ```

   Acceptance criteria: AC-FUN.3, AC-FUN.4

   Expected output: JSON formatted data containing elements "idUuid", "historyIdUuid", "mainData", "file", "result", "addedTime", "resultType", "mesage".

   Output:

   ```
   { "identity": {
       "idUuid": "6c7d30f6-269c-407e-a942-9855f7818386",
       "historyIdUuid": null,
       "mainData": {
         "lastName": "Smith",
         "firstName": "John",
         "age": 25
       },
       "file": "UEsDBAoAAAgAADeTiUq..." },
   ```

```
  "result": {
    "addedTime": "2017-04-09T15:27:03.361+0000",
    "resultType": "VALID",
    "message": "Signature is valid"}
}
```

Scenario step result: Successful

2. From step 1 output JSON data set extract field named "file" and convert with BASE64 decode tool to file named "scenario2.asice".

   Requirement: No requirement specified

   Input:

   ```
   BASE-64 encoded data from field "file":
   UEsDBAoAAAgAADeTiUqKIflFHwAAAB8AAAAIAAA.....
   ```

   Acceptance criteria: No acceptance criteria specified

   Expected output:

   Scenario step result: Successful

3. Open file "scenario2.asice" with Estonian digital signature tool "digidoc-tool" and check output.

   Requirement: SECREQ.1, SECREQ.2

   Input:

   ```
   digidoc-tool open --tslurl=https://demo.sk.ee/TSL/EE_T.xml
   --tslcert=trusted-test-tsl.crt scenario2.asice
   ```

   Acceptance criteria: AC-SEC.1, AC-SEC2

   Expected output: file with name "scenario2.asice"

   Output:

   ```
   Version
     digidoc-tool version: 3.12.3.1341
     libdigidocpp version: 3.12.3.1341_ddoc
   Container file: scenario2.asice
   Container type: application/vnd.etsi.asic-e+zip
   ```

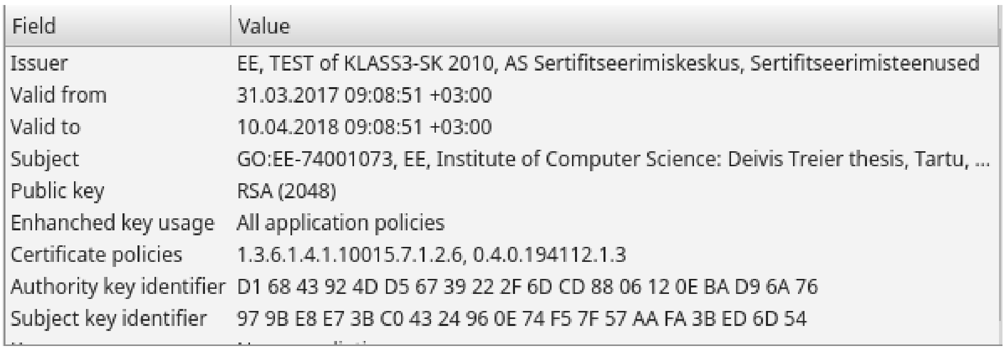
```
Documents (1):
  Document (application/json): data.json (61 bytes)
Signatures (1):
  Signature 0 (BES/time-stamp):
    Validation: OK
    EPES policy:
    SPUri:
    Signature method: http://www.w3.org/2001/04/xmldsig-more#rsa-sh
    Signing time: 2017-04-09T15:25:42Z
    Signing cert: Institute of Computer Science: Deivis
Treier thesis (TEST)
    Produced At: 2017-04-09T15:25:46Z
    OCSP Responder: TEST of SK OCSP RESPONDER 2011 (TEST)
    OCSP Nonce (20): 42 B4 21 45 CA 94 5F 95 9C 15 75 42
8B 1C 86 93 5F 45 FC E5
    TS: DEMO of SK TSA 2014
    TS time: 2017-04-09T15:25:45Z
    TSA:
    TSA time:
```

Scenario step result: Successful

4. By opening file "scenario2.asice" in Estonian digital signature program "Digidoc3.client" check Signer's Certificate under digital signature details view.

   Requirement: SECREQ.3, SECREQ.4

   Input: Input data is shown on figure 18



Field	Value
Issuer	EE, TEST of KLASS3-SK 2010, AS Sertifitseerimiskeskus, Sertifitseerimisteenused
Valid from	31.03.2017 09:08:51 +03:00
Valid to	10.04.2018 09:08:51 +03:00
Subject	GO:EE-74001073, EE, Institute of Computer Science: Deivis Treier thesis, Tartu, ...
Public key	RSA (2048)
Enhanched key usage	All application policies
Certificate policies	1.3.6.1.4.1.10015.7.1.2.6, 0.4.0.194112.1.3
Authority key identifier	D1 68 43 92 4D D5 67 39 22 2F 6D CD 88 06 12 0E BA D9 6A 76
Subject key identifier	97 9B E8 E7 3B C0 43 24 96 0E 74 F5 7F 57 AA FA 3B ED 6D 54

Figure 18: Validation artefacts and connections with requirements

   Acceptance criteria: AC-SEC.3, AC-SEC.4

Expected output: Field "issuer" contains Certificate type and issuer name "AS Sertifitseerimiskeskus" and Filed Public key contains at least RSA (1536)

Output: Field "Issuer" contains EE, TEST of KLASS3-SK 2010 what is part of Estonian Public Key certification infrastrucutre. Field "Public key" contains RSA (2048), what is more than minimum requirement.

Scenario step result: Successful

SCEN3 - Scenario for validating data tampering detection

1. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/full. (Positive flow when database table has correct hash link.)

   Requirement: SECREQ.9, FUNREQ.6, FUNREQ.7

   Input:

   ```
   curl -X GET --header 'Accept: application/json' \
      'http://audit-server:8080/validation/full?table_name=main_data'
   ```

   Acceptance criteria: AC-SEC.9, AC-FUN.6, AC-FUN.7

   Expected output: 1) Syslog file of Database Management System contains record about executed table full tampering detection. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

   Output:

   ```
   Syslog rows added to Database Management System log file:
   Executed table 'main_data' full tampering detection

   HTTP GET request response:
   {
     "addedTime" : "2017-04-14T20:31:39.995+0000",
     "resultType" : "VALID",
     "resultTableName" : "main_data",
     "message" : "Hash link is correct"
   }
   ```

   Scenario step result: Successful

2. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/-partial. (Positive flow when database table has correct hash link.)

Requirement: SECREQ.10, FUNREQ.6, FUNREQ.7

Input:

```
curl -X GET --header 'Accept: application/json' \
    'http://localhost:8080/validation/partial?
tableName=main_data&firstId=10&lastId=1500&initialHash=
32b5458065ef3bcd96dbc7efd3eaf705fcc03b83b91ace4a41fcdc09d2c277e0'
```

Acceptance criteria: AC-SEC.10, AC-FUN.6, AC-FUN.7

Expected output: 1) Syslog file of Database Management System contains record about executed table partial tampering detection. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

Output:

```
Syslog rows added to Database Management System log file:
Executed table 'main_data' partial tampering detection

HTTP GET request response:
{
  "addedTime": "2017-04-16T10:47:49.828+0000",
  "resultType": "VALID",
  "resultTableName": "main_data",
  "message": "Hash link is correct"
}
```

Scenario step result: Successful

3. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/-partial. (Positive flow when database table has correct hash link.)

   Requirement: SECREQ.11, FUNREQ.6, FUNREQ.7

   Input:

```
curl -X GET --header 'Accept: application/json' \
  'http://localhost:8080/validation/errorDetection
  ?tableName=main_data&firstId=10&lastId=1500&initialHash=
  32b5458065ef3bcd96dbc7efd3eaf705fcc03b83b91ace4a41fcdc09d2c277e0'
```

   Acceptance criteria: AC-SEC.11, AC-FUN.6, AC-FUN.7

Expected output: 1) Syslog file of Database Management System contains record about executed table partial tampering detection with error reporting. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

Output:

```
Syslog rows added to Database Management System log file:
Executed table 'main_data' partial tampering detection with
error reporting

HTTP GET request response:
{
  "addedTime": "2017-04-16T19:36:28.015+0000",
  "resultType": "VALID",
  "resultTableName": "main_data",
  "message": "Hash link is correct"
}
```

Scenario step result: Successful

4. By using PostgreSQL database client tool make SQL query to Database Management System table "main_data" to manipulate data in table record.

   Requirement: No requirement specified.

   Input:

   ```
   UPDATE public.main_data
     SET history_id_uuid='e5bb1ba3-302d-460b-a5b2-74e52034991f'
     WHERE id=1000;
   ```

   Acceptance criteria: No acceptance criteria specified.

   Expected output: Data is changed in table "main_data".

   Output:

   Scenario step result: Successful

5. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/full. (Negative flow when database table has incorrect hash link.)

   Requirement: SECREQ.9

   Input:

```
curl -X GET --header 'Accept: application/json' \
   'http://audit-server:8080/validation/full?able_name=main_data'
```

Acceptance criteria: AC-SEC.9

Expected output: 1) Syslog file of Database Management System contains record about executed table full tampering detection. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

Output:

```
Syslog rows added to Database Management System log file:
------Executed table 'main_data' full tampering detection

HTTP GET request response:
{
  "addedTime": "2017-04-16T19:38:26.381+0000",
  "resultType": "INVALID",
  "resultTableName": "main_data",
  "message": "Hash link is incorrect! Calculated: 3e6764c094110eaf0
last known: b7799266513250ee194b8b7182ae360ae53bdc8a046cc468377563
}
```

Scenario step result: Successful

6. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/-partial. (Negative flow when database table has incorrect hash link.)

   Requirement: SECREQ.10

   Input:

```
curl -X GET --header 'Accept: application/json' \
   'http://audit-server:8080/validation/full?able_name=main_data'
```

   Acceptance criteria: AC-SEC.10

   Expected output: 1) Syslog file of Database Management System contains record about executed table partial tampering detection. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

   Output:

```
Syslog rows added to Database Management System log file:
Executed table 'main_data' partital tampering detection
```

```
HTTP GET request response:
{
  "addedTime": "2017-04-16T19:39:29.054+0000",
  "resultType": "INVALID",
  "resultTableName": "main_data",
  "message": "Hash link is incorrect! Calculated: 54a9473602fa98382
last known: d777e9b4374e32f728afb7092fb44de0df17891ea0fbf1c067a19d9
}
```

Scenario step result: Successful

7. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/-partial. (Negative flow when database table has incorrect hash link.)

   Requirement: SECREQ.11

   Input:

```
curl -X GET --header 'Accept: application/json' \
'http://localhost:8080/validation/partial?tableName=main_data&
firstId=798&lastId=1001&initialHash=
3943393139905d38b006f75a165b6d93bfbe858a87b12c6cd68dfffd0314bc3a'
```

   Acceptance criteria: AC-SEC.11

   Expected output: 1) Syslog file of Database Management System contains record about executed table partial tampering detection with error reporting. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

   Output:

```
Syslog rows added to Database Management System log file:
Apr 27 21:12:55 database-server hash_audit: Executed table
'main_data' partial tampering detection

HTTP GET request response:
{
  "addedTime": "2017-04-27T21:12:55.056+0200",
  "resultType": "INVALID",
  "resultTableName": "main_data",
  "message": "Hash link is incorrect! Calculated:
   f24aa07feb5e2787a189746d01682fb524d40a99387b8d4f1bb49c60fbcec42
last known:
   e36f38b4acbb59a8e7bf495d822b2311a546734b8ffdb0f2846ede277c9e188
}
```

Scenario step result: Successful

8. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /validation/errorDetection. (Negative flow when database table has incorrect hash link.)

   Requirement: SECREQ.11

   Input:

```
curl -X GET --header 'Accept: application/json' \
'http://localhost:8080/validation/errorDetection?tableName=main_dat
3943393139905d38b006f75a165b6d93bfbe858a87b12c6cd68dfffd0314bc3a'
```

   Acceptance criteria: AC-SEC.11

   Expected output: 1) Syslog file of Database Management System contains record about executed table partial tampering detection with error reporting. 2) HTTP get requests retrieves JSON formatted data with tampering detection process results.

   Output:

```
Syslog rows added to Database Management System log file:
Apr 27 21:17:04 database-server hash_audit: Executed table
'main_data' partial tampering detection with error reporting

HTTP GET request response:
{
  "addedTime": "2017-04-27T20:17:04.435+0200",
  "resultType": "INVALID",
  "resultTableName": "main_data",
  "message": "Hash link is incorrect!
  Calculated: e5bb1ba3-302d-460b-a5b2-74e52034991f|b87957348fdca94
last known:
  e36f38b4acbb59a8e7bf495d822b2311a546734b8ffdb0f2846ede277c9e1886"
}
```

   Scenario step result: Successful

9. By using REST client or Swagger UI provided by solution make HTTP GET request with data to Information Application System API endpoint /result/list-ByTableName.

   Requirement: FUNREQ.8

   Input:

   ```
   curl -X GET --header 'Accept: application/json' \
     'http://audit-server:8080/result/listByTableName?table_name=main_
   ```

   Acceptance criteria: AC-FUN.8

   Expected output: List of database tampering detection results.

   Output:

   ```
   [
     {
       "addedTime": "2017-04-14T20:30:49.994+0000",
       "resultType": "VALID",
       "resultTableName": "main_data",
       "message": "Hash link is correct"
     },
     {
       "addedTime": "2017-04-14T20:31:08.983+0000",
       "resultType": "VALID",
       "resultTableName": "main_data",
       "message": "Hash link is correct"
     }
   ]
   ```

   Scenario step result: Successful

SCEN4 - Scenario for validating cryptographic link generation

1. **Requirement:** SECREQ.5, SECREQ.6, SECREQ.7

   Input: Source code in Appendix I.

   Acceptance criteria: AC-SEC.5, AC-SEC.6, AC-SEC.7

   Expected output: There is algorithm structure what deals with data acquisition from last database record

   Output: Acceptance criteria is met.

   Scenario step result: Successful

III. Solution application instruction details

Information Application System - Main application

Building - For building this component Maven is prerequisite to this task: run command in module source directory.

```
$ mvn clean package -Dmaven.test.skip=true
```

Deployment and configuration - Java container file "ais.jar" produced in last step must be placed into class path. Configuration file must be placed into class path directory config/application.properties

Running application - for running application use above mentioned command or run it inside system script.

```
$ sudo java -jar ais.jar \
    --spring.config.location=config/application.properties
```

Database Management System - Cryptographic operations shared object library

Building - in source code directory run following command

```
$ make
```

Deployment and configuration - Shared object library "hashchain.so" produced by last step must be moved into file system directory "/usr/lib/postgresql/9.6/lib/". After that PostgreSQL server must be reloaded (not reboot).

Database Management System - Database structure

Deployment - Database creation script is placed in file "database_create.sql" in source code catalogue.

Database Management System - syslog-ng

Configuration - Database Management specific configuration file "10-hashlog.conf" must be created into file system path "/etc/syslog-ng/conf.d".

```
# Audit Appliatoin System host configuration
destination d_syslog_tcp_audit {
    syslog("audit-server-ip" transport("tcp") port(2010));
};

# Log configuration to forward into file
log {
    source(s_src);
    destination(d_local);
};

# Log configuration to forward to Audit Application Ssystem
log {
    source(s_src);
    destination(d_syslog_tcp_audit);
};
```

Audit Application System - syslog-ng Java module

Building- For building this component Gradle is prerequisite to this task: run command in module source directory

```
$ gradle fatJar
```

Deployment and configuration - Copy h2writer.jar into file system

Running developed application - Java module is started on syslog-ng start.

Audit Application System - syslog-ng

Building - Build according to instructions in syslog-ng manual or install syslog-ng from package manager, but ensure, that java modules are installed.

Deployment and configuration - Sample syslog-ng Java module configuration is provided as follows. Configuration file "hash_chain.conf" must be created into file system path "/etc/syslog-ng/conf.d".

```
@module mod-java

# Configure listening server and port
source s_network {
 tcp(ip(0.0.0.0) port(2010));
};

# Configure to detect cryptographic meta data message
filter f_hash_chain {
 message("(hashChainAudit:)" type("posix"))
};

# Configure Java module "H2Writer"
destination d_h2_db{
   java( class_name("H2Writer")
   class_path("/module_location/h2writer.jar:
    /home/deivis/syslogng_to_h2/lib/h2-1.4.194.jar")
   option(
     "dbConnectionUrl",
     "jdbc:h2:file:/database_location/hash_messages;
        AUTO_SERVER=TRUE;DB_CLOSE_ON_EXIT=FALSE")
   option("dbUser", "writer")
   option("dbPassword", "writer")
   );
};

# Log configuration
log {
    source(s_network);
    filter(f_hash_chain);
    destination(d_h2_db);
};
```

Running developed application - For running syslog-ng in console use following command or application as system service..

```
sudo /usr/sbin/syslog-ng -Fe \
    --cfgfile=/etc/syslog-ng/syslog-ng.conf
```

Audit Application System - Database tampering detection management software (audit)

Building For building this component Maven is prerequisite. Run following command in source code directory

```
$ mvn clean package -Dmaven.test.skip=true
```

Deployment and configuration - Java container file "audit.jar" produced in last step must be placed into class path. Configuration file must be placed into class path directory config/application.properties

Running developed application - To start application run following command or application as system service.

```
$ sudo java -jar audit.jar \
    --spring.config.location=config/application.properties
```

IV. Licence

Non-exclusive licence to reproduce thesis and make thesis public