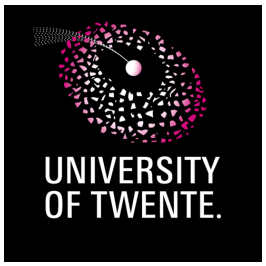


Universiteit Twente - Nedap N.V.

Computer Science with specialisation in
Cyber Security

Master Thesis Report

Bring Your Own Authenticator/Authentication Security in Physical Access Control Systems

Author:
Uraz Odyurt

UTwente supervisor:
Dr. Andreas Peter

Nedap N.V. supervisors:
ir. Albert Dercksen
Dr. ir. Fei Liu

22nd August 2016

Abstract

This work focuses on providing an efficient methodology for threat modelling of complex systems in their architectural design phase. The methodology, *architectural threat analysis*, along with new concepts for Bring Your Own Device (BYOD), specific to access control systems are provided. Scenarios based on the definitions of these new concepts, *Bring Your Own Authenticator (BYOAuthenticator)* and *Bring Your Own Authentication (BYOAuthentication)*, have been incorporated as the basis of an architectural threat analysis for the usage of mobile devices in *Physical Access Control Systems (PACS)*.

Throughout the conduct of such an analysis, a combination of different threat modelling tools, namely *attack trees* and *STRIDE threat lists*, have been considered in an iterative fashion. The resulting detailed, step-by-step analysis, reveals high-level threats and relevant mitigation considerations.

The study contributes to *secure-by-design* concept by providing an efficient and repeatable high-level architectural threat analysis methodology, as well as reusable BYOD terminologies, BYOAuthenticator and BYOAuthentication, including scenarios based on them. Architectural constructs based on these scenarios for a PACS, involving BYOD and biometrics, followed by their threat analysis, is a first and can be considered as a foundation for future studies.

Keywords: Architectural threat analysis, Threat modelling, Physical Access Control System (PACS), Biometrics, Access control, Bring Your Own Authenticator (BYOAuthenticator), Bring Your Own Authentication (BYOAuthentication), Bring Your Own Device (BYOD)

Acknowledgements

I would like to use this opportunity to express my gratitude to my supervisor from the University of Twente, Dr. Andreas Peter, for turning a lengthy study into an interesting challenge. I also would like to thank my supervisors from Nedap N.V., Dr. ir. Fei Liu and ir. Albert Dercksen, for providing me with an intellectual environment and constant support. Without their contributions, having a fruitful experience and achieving a satisfactory result would have been highly unlikely.

Contents

List of Figures

List of Tables

Abbreviations

AaaS Authentication as a Service

AD Auxiliary Data

ADTree Attack-Defence Tree

BAG Bayesian Attack Graphs

BCS Biometric Cryptosystems

BDMP Boolean logic Driven Markov Processes

BioAaaS Biometric Authentication as a Service

BYOAuthentication Bring Your Own Authentication

BYOAuthenticator Bring Your Own Authenticator

BYOD Bring Your Own Device

CB Cancelable Biometrics

DFD Data Flow Diagram

DoS Denial-of-Service

FHE Fully Homomorphic Encryption

FICAM Federal Identity, Credentialing and Access Management

HCE Host Card Emulation

ICT Information and Communication Technology

ID Identity

IEC International Electrotechnical Commission

IP Internet Protocol

ISO International Organization for Standardisation

JVM Java Virtual Machine

MDM Mobile Device Management

NFC Near Field Communication

OWASP Open Web Application Security Project

PACS Physical Access Control System

PI/PI* Pseudo Identity

PIC Pseudo Identity Comparator

PIE Pseudo Identity Encoder

PIN Personal Identification Number

PIR Pseudo Identity Recoder

PIV Pseudo Identity Verification

PKI Public-Key Infrastructure

QAA Quality Authentication Assurance

SaaS Software as a Service

SaaSS Service as a Software Substitute

SD Supplementary Data

SDL Security Development Lifecycle

SIA Security Industry Association

STORK Secure idenTity acrOss borDers linKed

STRIDE Spoofing, Tampering, Repudiation, Information disclosure, Denial-of-service, Elevation of privilege

SWHE Somewhat Homomorphic Encryption

UML Unified Modeling Language

USB Universal Serial Bus

1 Introduction

Nowadays, in every aspect of Information and Communication Technologies (ICT) we are facing the exponential expansion of personal mobile device usage. This phenomenon, in whichever scenario and use-case, is being addressed by a single umbrella term, *Bring Your Own Device (BYOD)*. There are numerous benefits and drawbacks, but the important fact to consider is that these benefits and drawbacks are not the same for different applications and use-cases, suggesting the necessity of more granular definitions. Take a *Physical Access Control System (PACS)* for instance. The way a mobile device may be used in a PACS environment, and in general, an access control environment, is completely different compared to the traditional remote access to organisational resources. Such specific utilisations demand special addressing.

Adding to the complexity and the special nature of our example, consider incorporation of biometrics with the access control system. To be more precise, because of the direct link between biometric readings and owners, there can be different interests to look into. Is it achievable to read biometric attributes in a one-way fashion, so that it would not be possible to trace and match a reading with its owner? Can we store biometric readings securely, so in case of data theft, readings would be useless to perpetrators? Is it possible to preserve owners' privacy by preventing the serving infrastructure from gaining information about them, while processing requests? All these and more are valid angles and need to be addressed. Especially, the privacy preserving aspects are highly sought-after.

Taking the point a step further, when it comes to biometrics and access control, concerns regarding security and privacy overlap. This is the result of both identifying and verifying potential of biometrics. We believe the best way to understand security and privacy concerns, relevant to specific use-cases of a technology, is to conduct *architectural threat analysis* for those specific use-cases. That is our aim throughout this report, in a nutshell, to consider a PACS environment, involving mobile devices as BYOD and utilising biometric identification/verification for access control.

As ICT systems evolve into more complex implementations and include a mixture of network communication, software processes, protocols, third-party services, mobile and static nodes, etc., there is a serious need for threat analysis methodologies. When it comes to software processes, the benefits of threat modelling is well-known. Threat modelling as a part of any security concious best-practice, such as Security Development Lifecycle (SDL) [19], only focuses on software development aspects. Using such a practice in combination with other information, could provide us with an analysis on a higher and more abstract level. We shall call this *architectural threat analysis* and as the name suggests, it is applied to architectural designs.

Applying threat analysis in such levels could provide developers and system architects with viable design options, allowing them to choose a fitting solution based on their requirements specification. The result of this process is not necessarily a single correct solution, but a number of variations for different requirements and use-cases. A high-level architectural threat analysis practice will make solution development efficient and will bring in the security conciousness early on, contributing to a *secure-by-design* solution.

1.1 Related work

There has been different efforts to provide a generic approach for threat modelling, such as [22] and [46]. The clear fact is that there is no unanimous agreement on how to approach the

practice. Expectedly, the understanding about threat modelling mostly focuses on software development. Rhee et al. [31] also provide a rather structured approach towards threat modelling, but it seems to be case specific. Our goal is to improve these efforts by providing a repeatable methodology and base the threat modelling on the architectural design and a pool of scenarios. There has also been developments in the actual tools used for threat modelling processes [3, 9, 17, 26], although, we find a combination of attack trees and STRIDE lists more suitable for the iterative nature of threat modelling and we utilise these in our process.

One of the only relatively dependable starting points for PACS architectures is the white paper [39] from Security Industry Association (SIA) [37]. We have heavily complemented architectures using Nedap's internal knowledge and given scenarios, involving biometrics and mobile devices. When it comes to threat analysis of biometrics as part of a more complex system, sources [28, 56] have gone through the major attack vectors, but they all focus on biometric specific attacks and defences, rather than a threat analysis of the system.

1.2 Contribution

By improving and combining threat modelling practices, attack trees and STRIDE threat lists, we take the idea of threat modelling for software development and evolve it to an *architectural threat analysis*, which is more high-level and intended for architectural design phase. In this fashion, the *secure-by-design* property can be fulfilled more efficiently and early on. We also apply this architectural threat analysis methodology on PACS implementations involving mobile devices and biometrics as a first for such a combination. The results can be utilised for future designs based on these technologies and they can also be considered as starting points for other access control scenarios.

The notion of BYOD is a far reaching one. Mobile devices are not used in the same fashion for every task and in every architecture. We propose two special cases, *Bring Your Own Authenticator (BYOAuthenticator)* and *Bring Your Own Authentication (BYOAuthentication)*, in relation with access control systems. Furthermore, we consider both cases in a PACS environment, where biometrics is used for identification/verification.

Based on our study, there has not been any academic research focusing specifically on PACS, let alone its combination with other technologies, namely, biometrics and BYOD. Based on the definitions of BYOAuthenticator, BYOAuthentication and generic PACS practices, we provide architectural variations for the combination of *BYOD-PACS-biometrics*. We will also carry out architectural threat analysis of these variations, pointing out security implications related to them. We specifically point out mitigation possibilities in such environments, based on biometric template storage protection schemes. Knowing these potential threats and mitigations, designers can choose the right solution to fulfil the intended requirements specification.

1.3 Structure of the report

The rest of this report is structured in the following fashion. The research approach is detailed in Section 2, including the research question and methodology. Thereafter, Section 3 introduces main concepts and terminologies to be used throughout the report. Formalisations of relevant BYOD categories, as well as trust are included in Section 4. Section 5 goes through the architectures and scenarios involving both static and mobile readers in a PACS environment. Section 6 presents the threat identification process using a mixture of threat modelling methods,

complemented by extra definitions. This part includes the bulk of architectural threat analysis study in this report, followed by Section 7 and Subsection 7.6, mentioning results and mitigation possibilities, respectively. Section 7 can be considered as a discussion ground for the mitigation potential of biometric template protection methods, introduced in Subsection 3.4. The report is concluded in Section 8.

2 Outline of the research

In this section, we will go through the different stages of the research, as well as the initial motivation for it. The incorporated methodology and other bits and parts will be presented, drawing a holistic picture.

2.1 Research question

Based on what we know so far and the available literature, as well as the experience within the industry, we can formulate the following research question.

> *Is there an efficient and repeatable approach to apply an architectural threat model analysis on complex ICT designs? Based on such an analysis of Physical Access Control Systems (PACS) and considering the involvement of mobile devices as Bring Your Own Authenticator (BYOAuthenticator) and Bring Your Own Authentication (BYOAuthentication), which components of the system are weakest links? What are the most important attack vectors for these components and how can they be mitigated?*

2.2 Methodology

This study can be divided into two major parts. In the first part, the goal is to perform threat analysis. We will use two major threat modelling tools, attack trees and STRIDE lists, in combination. Attack trees and STRIDE lists are both tools for methodical categorisation of threats applicable to a given system. In practice, these tools are used as replacements of each other, but we will consider them as different steps in our architectural threat analysis. To be able to come up with a sensible result, we start from the known and move towards the unknown. This means that initially, we use the results available through literature, where threat analysis of systems along with their known attack vectors are introduced. For instance, there are good analyses already available for abstract biometric systems [28, 56], since every biometric system is made up of more or less standard elements. This will be the known part and it is based on threat modelling via attack trees. As for the unknown part, we will create STRIDE threat modelling tables based on the attack vectors from the previous step (leafs from the attack tree), but we will consider them for reference architectures of PACS. *Data Flow Diagrams (DFD)* generated from these PACS architectures will be the basis of this linkage.

For the most part, our analysis is based on the placement of *temporary artefacts*, *persistent artefacts* and *the verification process* itself. This is a result of the fact that *temporary artefacts* (e.g. biometric sample) and *persistent artefacts* (e.g. biometric template) are crucial elements in biometric systems. By induction, *the verification process*, handling both of these, is also

important. Although these notions are elaborated later on, but it is worth mentioning that temporary and persistent artefacts are artefacts involved in the verification process, differing based on their lifecycle. A deliverable is temporary to the system, while a persistent artefact is stored in the system. We will consider these elements and their placement in PACS architectures involving BYOD. The BYOD spectrum is also narrowed down to BYOAuthenticator and BYOAuthentication definitions. In turn, four concrete scenarios emerge from these definitions, which we consider for our analysis and results.

Later on as the second part of our study, we look into mitigation possibilities, resulting from biometric template protection schemes. A short state-of-the-art for biometric template protection schemes will also be included as background information. In short, we can list the steps in the following enumeration, visualised in Figure 2.1.

1. Functionality-based, high-level threat modelling, considering the elements of a generic biometric system, using attack trees.

2. Architectural threat modelling using STRIDE lists for leafs of the attack trees, considering the introduced PACS architectures and DFDs based on them.

3. Mapping of attacks to BYOAuthenticator and BYOAuthentication scenarios and discovering challenges.

4. Studying mitigation possibilities for attack vectors, based on biometric template protection schemes.

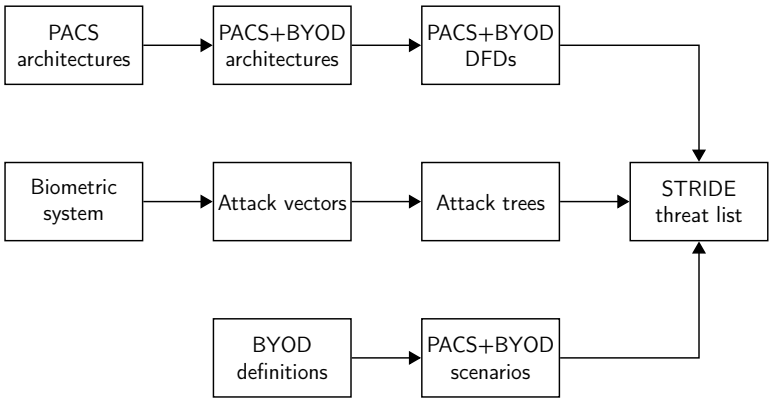


Figure 2.1: Steps of the methodology

2.3 Limitations and scope

As with any research, there are limitations to be taken into consideration. Threat analysis and threat modelling methods are primarily used for evaluation and analysis of software implementations, whereas here, we are using them for high-level analysis of system architectures at design stage. This introduces challenges in the form of uncertainties and from time to time, we must compensate for them by means of definitions and assumptions. Another limitation is the fact that the architectural designs for PACS are highly variable. This is due to the specific requirements for each implementation and a direct result of the highly flexible nature of

PACS, allowing the architecture to include, or exclude different components. We try to provide reference architectures, covering most common implementations and set-ups. Specifically, we assume an architecture involving readers, controllers and central servers as a generic starting point.

Following the above considerations, our aim is to demonstrate the effectiveness of the introduced methodology and as a generic example, we will apply it to scenarios derived from fundamental decisions, such as using BYOD and biometrics. We will keep the process as abstract as possible, for repeatability of the methodology is an important factor. Also, when it comes to threat analysis, we are not focusing on statistical or quantitative risk evaluation, as such studies move away from generality.

3 Background and main concepts

This section includes main concepts and terminologies, relevant throughout this report and required for the initial familiarity of readers. These concepts are standard terminologies and definitions in computer science. New introductions and terminologies by the author are included in Section 4. Some of these concepts are not used as frequently as the rest, but they play a familiarising role nevertheless.

3.1 Physical Access Control System (PACS)

A Physical Access Control System (PACS) can be thought as a combination of three definitions, taken from the Internet Security Glossary [40], *access*, *access control* and *physical security*.

Access The initiating entity will have the means of communication, or interaction with a target, having the intention of system resource usage, information manipulation, or gaining insight about the stored information at the target [40].

Access control Regulation of system resource usage and limitation of access to these resources, based on a security policy, resulting in access only by authorised entities. These entities include users, applications, processes, etc. and are defined in the security policy [40].

Physical security Physical security is the practice of preventing physical access to systems/resources through physical measures. Locks, walls, alarms and so forth are examples of these measures [40].

3.1.1 Overview

A PACS is the combination of the aforementioned definitions, access, access control and physical security. In short, a PACS provides access control for organisational resources, based on the organisational security policy and through physical measures. A PACS provides authentication, authorisation, administration, auditing (the four "A"s) and more recently, analytics (making it five "A"s) [39]. PACS are networked information technology systems and consequently, Moore's Law applies to them. This means that price-performance ratio constantly improves and new technologies are introduced regularly [39].

The actual architectures and included components in a PACS are highly dependant of the case scenario at hand. There are no standards describing a generic architecture as a best-practice. Nonetheless, to have a point of reference, we will use SIA's white paper [39], titled "Physical Access Control System (PACS) in a Federal Identity, Credentialing and Access Management (FICAM) Framework", as well as Nedap's internal knowledge and expertise for our work. The combination of both resources will lay the foundation for our analysis. Obviously, we will add, or remove components and functionalities to arrive at a technology and architecture agnostic design. We have to reiterate that the components and architectures provided in Subsection 3.1.2 and Section 5, respectively, are not academically binding. The aim of such architectural brainstorming is to help us with the visualisation and to result in a better understanding of the interactions between different components.

3.1.2 Components and communication types

There are different components commonly used in PACS implementations. Here, we will go through the most essential ones in our assumed PACS implementation involving biometrics. As a reiteration, component usage in real-world implementations depends on the use-case at hand. There are other industry standards available describing PACS components, such as "ONVIF Profile C Specification" [24], which are out of the scope of this study.

Central server The central server in a PACS can be considered as the back-end for the rest of services and components. Generally speaking, a central server includes three layers, user interface, business layer and a lower layer made up of daemon services, communicating with other entities, such as controllers. The central server is used by different administrators with different levels of privilege. Access control rules are also defined in central server. A central server also includes registration and policy management capabilities [39].

Database PACS database directly communicates with the business layer of the central server. Any data that needs to be stored in a persistent fashion is kept in the database.

Controller Controllers, or field controllers as they are called by SIA [39], are capable embedded devices, communicating with readers, doors and the central server. They usually run a GNU/Linux distribution, along with necessary modules such as applications written in C for instance, or Java Virtual Machine (JVM) for cases of applications written in Java. These embedded devices can also be programmed microcontrollers, eliminating the need for an operating system. Decision making on granting or denying access occurs here. Distributed intelligent controllers, as their name suggests, can be deployed in distributed architectures when the size of the implementation requires it.

Transparent readers Transparent readers, or simply readers, do not perform any interpretations and are not involved in decision making [39]. These readers only relay information between controllers and cards, or between controllers and biometric attributes.

Intelligent readers Intelligent readers can perform additional tasks, such as interpretation of commands and responses exchanged with cards, or performing template-sample comparison for biometrics, amongst other tasks.

Doors A door in PACS terminology can be broken down to three parts, the door itself, locking mechanism and door interface. The door interface is used for communication with readers and controllers depending on the set-up, and is considered the link between the actual door and PACS.

Connectivity Communication medium between components can be a rather diverse set of technologies. For the purpose of this study, we will only consider Internet Protocol (IP) and Serial lines. Usually the connection from controller to central server is IP-based and the connection from controller to readers and doors is over serial. Other types of connectivity include different wireless standards for communications between smart cards, or mobile devices and readers, such as Near Field Communication (NFC).

Biometric verifier A biometric verifier can be thought as a controller, or a central server specifically for biometric readers. Biometric verifier can control biometric readers and receive samples taken. Samples can be compared with templates, using biometric verifier's matcher, and the database storing these templates can be a part of the biometric verifier.

Template storage Template storage is a database of biometric templates, which can be a part of biometric verifier, or can be on a separate machine. The important point is that when we say template storage, in general we mean a central database. Although biometric templates stored on cards, or mobile devices could also be covered in this terminology.

The following are not physical components, but virtual mechanisms and processes, which may run on different physical components.

Identification and Authentication There can be different combinations based on different identification and authentication mechanisms. Identification could be done using PIN, physical identifier (card), or biometrics. Identification will be combined with an authentication method, but the general practice in the industry does not involve authentication at all, when using biometrics for identification. With card-based identification, authentication can use biometrics verification, certificate verification, or PIN verification.

Authorisation The authorisation process follows after authentication, with the goal of authorising access to resources based on the defined policy.

3.2 Biometrics

According to Jain and Ross [14], biometrics is the science of identifying individuals, using their physical, chemical or behavioural attributes. Biometrics as something you are, is mainly seen as a convenient improvement over something you know methods, such as passwords, or token-based methods, such as ID cards. Despite claims by some literature [14], arguing in favour

of added security through biometrics, a biometric system has its own attack vectors like any other system. Also considering the privacy sensitive nature of biometric measurements, one can easily see the higher risk involved with compromising biometric measurements. However, functionality-wise, biometrics offers *negative recognition* and *non-repudiation* [27], which are not provided by passwords, or tokens. Negative recognition is to detect an individuals already enrolled status and prevent the individual from multiple enrolments. This is useful for welfare allocation for instance. With non-repudiation, individuals cannot deny their access later on.

Biometric attributes, also known as traits, indicators, identifiers, or modalities are numerous. Commonly used attributes include, face, fingerprint, hand geometry, palm print, iris, keystroke pattern, signature, voice and gait. The complete list is more extensive, including ear, hand vein, odour or the DNA information [12, 54]. No matter which kind is used, ideal biometric attributes should fulfil a number of requirements. This is not the case in real world and each attribute has its own strong points and will be chosen based on the requirements of the use-case. The following is the seven ideal factors used in assessing suitability of a biometric attribute [14].

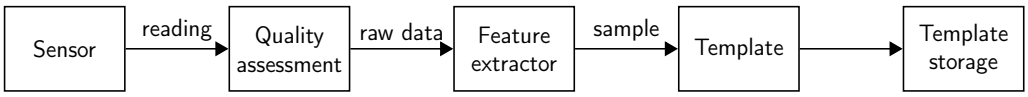
1. Universality: Every individual can provide the biometric attribute.
2. Uniqueness: There is enough differentiation between attributes of individuals.
3. Permanence: The attribute does not change over long periods of time, or the change is insignificant to the matching process.
4. Measurability: It is possible to capture and process the attribute in digital realm.
5. Performance: Accuracy and resource consumption is acceptable.
6. Acceptability: Individuals accept to use and provide the attribute.
7. Circumvention: The attribute is hard to be faked.

A generic biometric system includes a number of modules, namely, sensor module, quality assessment and feature extractor module, matching and decision making module, and system database module [14]. The system also is capable of three different workflows, involving the aforementioned modules.

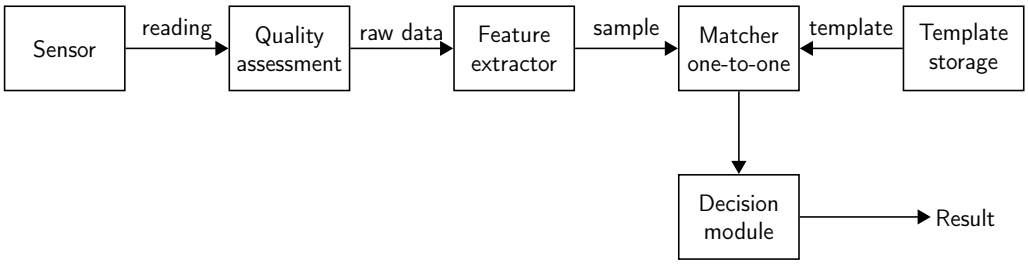
Enrolment This is the process of biometric template creation after collecting a feature set (sample) from an individuals biometric reading. The process is partly similar to biometric sample creation, except a template will be saved in a template storage and will be used later on to compare the biometric sample with. The generic workflow is depicted in Figure 3.1a [14].

Verification Biometric verification, which is synonymous with authentication in this report, is the process of verifying claimed identity of an individual by means of comparing a biometric sample to a biometric template. Biometric verification always involves an identification step prior to it, with a PIN code for instance. The identifying data allows the system to pinpoint claimed biometric template(s) in the biometric storage and fetch it for comparison. This way the sample is compare against returned template(s) and thus, the process is called *one-to-one* matching. The generic workflow is depicted in Figure 3.1b [14].

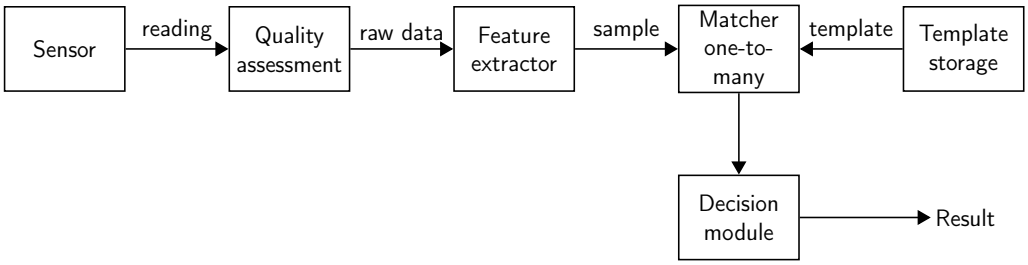
Identification Biometric identification uses a biometric sample as the identifying data. In this case, the sample needs to be compared to all templates available in the template storage. As such, the process is called *one-to-many* matching. The generic workflow is depicted in Figure 3.1c [14].



(a) Biometric enrolment workflow



(b) Biometric verification workflow



(c) Biometric identification workflow

Figure 3.1: Biometric system workflows

3.3 Security vs. privacy

The terms *security* and *privacy* and concepts surrounding them, for instance the methods used to provide security or privacy, has to be defined properly. According to the Internet Security Glossary [40], there can be a number of definitions depending on the area of application.

Security can be interpreted as the following.

- Recommended definition: "A system condition that results from the establishment and maintenance of measures to protect the system [40]."
- Recommended definition: "A system condition in which system resources are free from unauthorised access and from unauthorised or accidental change, destruction, or loss [40]."
- Recommended definition: "Measures taken to protect a system [40]."

Privacy can be interpreted as the following.

- Recommended definition: "The right of an entity (normally a person), acting in its own behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share its personal information with others [40]."

- Other definition: "The right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed [40]."

We can clearly see the meaning behind security is the protection of system state, functionality and resources, whereas privacy's meaning involves the protection of information related to persons. The latter definition also implies the owner's will as a requirement for access. When it comes to achieving such protections, there are different technical measures available and there are measures providing, or at least influencing both security and privacy at the same time. This also applies to processes. A compromise of certain processes will harm both security (protection of system) and privacy (protection of personal data). As a relevant example, malicious access to biometric data in an access control system such as a sample, or a template, is harmful to both security and privacy. For biometric data is both, information related to an individual (privacy) and information used for access control (security).

3.4 Template protection state-of-the-art

There has been an abundance of studies with the sole purpose of protecting sensitive biometric data, especially the biometric template. These studies have resulted in numerous template protection schemes. Accordingly, Breebart et al. [4], consider eight key requirements for biometric templates. The aim is to have a privacy-protected verification scheme.

1. Protected templates: Samples, feature and templates are protected from retrieval, decoding and being uniquely linked to subjects.

2. Revocable, renewable and diversifiable protected templates: A revocation mechanism for templates has to be in place, as well as diversification, making it possible to generate independent representations from biometric attributes.

3. Universal approach: Template generation should be applicable to different kinds of biometric attributes and even to combinations of attributes, to create higher definition and security.

4. Interoperability: Predefined formats and methods should be used to have interoperable output from sensors and feature extractors.

5. Data minimisation: The actual template data should be minimised in favour of privacy, storage and communication constraints.

6. Intrinsic security: Information regarding the amount of success for a failed attempt should not be available, unless justified.

7. Seamless integration with existing verification methods: Flexibility for integration with different knowledge-based, or possession-based authentication and 2 or 3-factor verification methods should be part of the scheme.

8. Architecture flexibility: Different architectures should be supported with the possibility of remote or local verification. A remote verification evolves a central template database.

3.4.1 Regular schemes

Based on Breebaart et al.'s [4] thorough definition of requirements for protected templates, one way to assure these requirements is to generate a unique piece of data from a biometric template, which can be revoked in case of a compromise. It should also be impossible to create the original template from this data. This piece of data is called a *Pseudo Identity (PI)* and the creation process involves a second part, *Auxiliary Data (AD)*. AD could be different depending on the method used in a scheme to generate PI. A PI generated during the enrolment process can be compared to a PI generated during a sampling process for verification. The AD used in both processes is the same. These terms were also used in ISO/IEC 24745 [11], defining biometric information protection requirements and guidelines. PI and AD are generated using *Pseudo Identity Encoder (PIE)* during the enrolment process. PIE accepts feature set of a sample from feature extractor as its input and generates PI and AD. PIE can also accept an extra input as *Supplementary Data (SD)*, which can be used for increased security for instance. An example can be knowledge-based secrets entered by the carrier, resulting a biometrically hardened passwords [20].

When it comes to the verification of carriers using such template structures, there are two different approaches, *PI Recoder (PIR)* and *PI Verification (PIV)*.

PI Recorder (PIR) The PIR approach is based on the recreation of PI during the verification process. A PIR accepts feature set and AD as its inputs, generating a PI*, to be compared to the original PI generated during the enrolment process. PIR can also accept SD if it has been used during the enrolment process. The comparison between the original PI from the enrolment process and PI* from the sampling process is carried out by *Pseudo Identity Comparator (PIC)*. This approach is intended for physically separate PIR and PIC and its main advantage is protected exchange of information between the two. For instance, PIR could reside on the sensor, while PIC is on the service provider side [4]. Most current template protection schemes can be explained using this approach. The approach is depicted in Figure 3.2.

PI Verification (PIV) The PIV approach does not recreate PI and instead, accepts the original PI from the enrolment process as an additional input. Thus, a PIR accepts feature set, AD and PI as its inputs, generating a final verification result. PIV can also accept SD if it has been used during the enrolment process. This approach is intended for physically unified PIV and template storage, as in such a case, there is no need for any transmission. For instance, a matching solution on a smart card, or any tamper-proof token, could utilise this approach [4]. The approach is depicted in Figure 3.2.

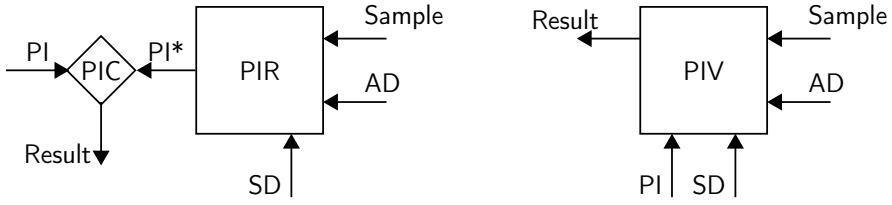


Figure 3.2: PIR and PIV verification approaches

Possible schemes following the aforementioned approachs include, fuzzy commitment [16],

cancelable biometrics [29], helper data systems [50], biometric encryption [44], fuzzy vault [15], shielding functions [18], fuzzy extractors [7] and extended PIR [5]. These schemes use different methods for PI generation and thus, use different AD. Some literature name AD as helper data in a collective fashion for all schemes. Table 3.1, partly taken from [4], elaborates PI and AD elements related to different these template protection schemes.

Template protection	**PI**	**AD**	**Verification**
Fuzzy commitment	hash of secret string	offset	PIR
Cancelable biometrics	transformed template	transform parameters	PIR
Helper data systems	hash of secret string	helper data	PIR
Biometric encryption	cryptographic key	filter and key link	PIR
Fuzzy vault	hash of secret string	point set P	PIR
Shielding functions	hash of secret string	authentication challenge W	PIR
Fuzzy extractors	hash of secret string	public string P	PIR
Extended PIR	encrypted template	n/a	PIV

Table 3.1: Template protection methods along with their relevant PI and AD elements

Rathgeb and Uhl [30] give another overview of template protection schemes, reiterating importance of biometric data storage risks for privacy aspects. They present a different categorisation, dividing template protection scheme into *Biometric Cryptosystems (BCS)* and *Cancelable Biometrics (CB)*.

Biometric Cryptosystems (BCS) BCSs aims at either binding a digital key to a biometric attribute in a secure fashion, or generating a digital key from a biometric attribute. Examples of key-binding schemes include, Biometric Encryption, fuzzy commitment and fuzzy vault. Examples of key-generation schemes include, private template and quantisation schemes. The major security issue with BCSs is the privacy leakage through the information contained in the helper data [30].

Cancelable Biometrics (CB) CBs follow the method of distorting biometric raw data by means of different transforms and perform sample-template comparison in the transformed domain. Two subcategories of CB are, non-invertible transforms and biometric salting [30]. It is a known fact that non-invertible transforms provide higher security, while having lower recognition performance, in comparison with biometric salting [13].

3.4.2 Homomorphic encryption

Another, more recent development in template protection schemes is to keep verification process execution in encrypted domain by using *homomorphic encryption*. Initially called *privacy homomorphism* [32], there has been a number of *partially homomorphic* schemes, based on RSA, ElGamal, Goldwasser–Micali, Benaloh and Paillier cryptosystems. Partial homomorphism only provides homomorphism for one type of operations, either addition, or multiplication. Gentry [8] has introduced the first *Fully Homomorphic Encryption (FHE)* scheme, supporting arbitrary additive and multiplicative (or basically any function) homomorphisms. In short, a

FHE scheme allows computations with arbitrary functions over encrypted data, but without the need for decryption [8]. This means that untrusted third-parties can be used for calculations, without compromising data security and privacy. Examples can be any cloud-based service, and in our case, remote verification services.

A number of open problems with FHE schemes have deemed them impractical at their current state. Issues such as susceptibility to lattice-based attacks, parameter selection complexity, inefficiency of implementations, the need for algorithmic optimisations, large parameter and huge cypher-text sizes, and finally, low performance and high memory consumption [21].

Another interesting construction is *Somewhat Homomorphic Encryption (SWHE)*, allowing for a limited number of additions and multiplications in the encrypted domain [55]. This is quite useful for biometrics, as verification algorithm can be executed using the available functionality in the encrypted domain. The approach is very similar to PIV, but both sample input and result output will be encrypted and can only be read using a private-key.

3.5 Mobile devices

Since mobile devices are an important part of this study, in this part we will define which exact subset of these devices are being considered. We will also give an abridged overview of mobile device security implications without going into details, as it is a broad topic on its own.

3.5.1 Mobile device definition

We are including it in our designs and discussions, so let us first define what we consider as a mobile device. The term could easily span a diverse spectrum of devices, from laptops to smartphones, to smart cards. Some might even say a USB storage is a mobile device. For the purpose of this report and throughout, we will consider a mobile device as a smartphone, or a similarly capable device. By this we mean a device, having similar processing, storage, user intractability and communication capabilities as a smartphone. We will especially differ a mobile device from a smart card and even consider comparing the two in different scenarios.

3.5.2 Mobile device security overview

History has proven that security through obscurity is a futile practice. As Kerckhoffs' principle states, the security of a cryptosystem should not rely on the secrecy of anything, but the key. In other words, no matter how complex, one can always expect the system to be reverse engineered.

As mobile device development has a particularly fast pace, one must always concentrate on recent publications. At the same time, we cannot ignore the fact that, because of heavy adoption and huge numbers, there are so many mobile devices with outdated hardware and software in active use. This means no matter how much security aspects advance, it is not going to benefit a large number of these devices. Keeping that in mind, we will briefly touch upon the important factors regarding mobile device security and its implications from a PACS perspective, especially in combination with biometrics.

A third-party system running obscured security processes, simply cannot be trusted. This is a general fact, applied to any third-party service, or device. The risk is much higher in the case of mobile devices, precisely because of their mobile nature, portability and lack of physical

security. Let us give the example of a mobile device here. As much product and technology agnostic as we would like to be, but given the fact that the internal secure architecture design by ARM (called TrustZone [2]), including a secure storage, is the first concrete hardware security module design for mobile devices, the exemplification will be about a proprietary technology. Still, this will not be in vein, since we will also point out the limitations of a proprietary design.

When it comes to storing highly sensitive data on a mobile device, at the moment, the security of this data is as good as the security of the secure storage and the processes accessing it. In comparison with smart cards, a mobile device is autonomous and its internal functionality is obscured from the operator. Since a smartphone is highly capable, both in terms of processing power and diversity of communication channels, attacks are more various and more powerful.

ARM's TrustZone design is capable of running its own software, separate from the device operating system. TrustZone supports different software architectures and it can even run a dedicated operating system [2]. As an example, Apple's Secure Enclave, which is based on ARM's design, runs a dedicated microkernel of the L4 family [1]. The fact that these software architectures and their processes are proprietary creates the obscurity. No matter how separated these dedicated security processors are from the actual mobile device software and architecture, there exists a process to update the software/kernel running on them. Without thinking too much about other factors, this can be one attack vector and again, the update process is proprietary and there are no details published about it.

Overall, there are numerous publications around the topic of mobile device security and specifically smartphones, confirming many attack vectors and poor record of vulnerabilities and exploitations for these devices. Seacord [36] provides a compact categorisation of security sensitive perspectives for smartphones. These include malware inserted in applications and distributed through third-party app stores, data storage practices and consequently key storage security, telecommunication network security, mobile operating system security features and vulnerabilities, and secure coding practices. In addition to these, Vidas et al. [52] point out the processing power and communication channel diversity of smartphones, as well as large scale distribution of mainly three operating systems. Delving into Android as one of the most popular mobile operating systems, numerous flaws in Android security model and patch cycle, along with a taxonomy of attack classes are presented [52]. These facts depict a rather disturbing picture and reminds us of the human weak link, which is always under the risk of exploitation through social engineering.

To enforce enterprise control measures on employee smartphones, we can refer to Mobile Device Management (MDM) practices. MDM systems are used to manage, monitor and control smartphones in BYOD scenarios [31]. Rhee et al. [31] give a list of enforceable controls using MDM from application management to different security management tasks. The important thing to point out here is that, the only component guaranteeing the enforcement of these controls, is the MDM agent on the mobile device. Considering the fact that a MDM agent is basically an application with numerous privileges, residing on the application domain of the mobile device, it is still susceptible to the same attack vectors. The MDM threat modelling provided in [31], makes this abundantly clear.

Following the aforementioned security implications, we must remember that perfect security is an ideal endeavour and cannot be achieved in real-world situations. One should simply consider the risks when introducing mobile devices as part of a system's design. Major risks, or attack vectors summerised in US-CERT publication [34], Cyber Threats to Mobile Phones,

are as follows.

- Physical characteristics of mobile devices makes them susceptible to theft and loss, providing the attacker with unlimited amount of time.
- Unofficial and third-party software and apps may include malicious behaviour, as there is little or no security evaluation in places in most cases.
- Official software, including apps and operating system, running on mobile devices are susceptible to exploitation and include vulnerabilities. This is a side effect of mobile phone evolving to be as capable as personal computers.
- Numerous communication channels introduce different ways to perform phishing attacks and social engineering.

3.6 Authentication as a Service (AaaS)

To understand Authentication as a Service (AaaS), we first need to cover what Software as a Service (SaaS) model is. The main idea behind SaaS model is the differentiation of ownership and usage of software [49]. In that sense, software's goal is to deliver a service [49], which would make it much more accessible and scalable. On the other hand, there are huge concerns involved with SaaS model from security and privacy point of view. The issue has been best described by Richard Stallman. He puts forward an alternative name for the model, Service as a Software Substitute (SaaSS) [45] and continues,

> "With SaaSS, the users do not have even the executable file that does their computing: it is on someone else's server, where the users can't see or touch it. Thus it is impossible for them to ascertain what it really does, and impossible to change it [45]."

The same can be pointed out for the actual data being processed by the service and kept on the cloud.

Today, any cloud-based service is designed around SaaS model. Services provide functionality and a service providing authentication functionality under SaaS model is considered Authentication as a Service (AaaS). Consequently, a generic AaaS set-up as shown in [38] involves configuration data, authentication logic, authentication data and API as its components.

It is important to remember that within the scope of this report, we are not dealing with the generic AaaS in our architectural analyses, but our focus is specifically on Biometric Authentication as a Service (BioAaaS). Accordingly, the specific requirements and characteristics of biometric authentication are the factors influencing our results.

3.7 Threat modelling

Any form of analysis intended for a complex interconnecting system requires a structured approach. Failing to follow such a practice will seldom produce tangible and useful results. Our focus here is a generic PACS. Like any other complex system, a PACS involves different software and hardware components and subsystems, as shown in Subsection 3.1.2.

Threat modelling is a systematic approach towards finding security problems and vulnerabilities in a software. We say software, since the practice is intended for software development and interconnecting software systems in particular, such as web applications. Nonetheless, we will use it for threat analysis of a PACS and we will do it with a high-level perspective. Threat modelling can have different flavours, focusing on different aspects such as, attacker perpective, asset perspective, or system perspective. We will follow the latter, requiring an architectural study of the system. The steps involved in the process are shown in Figure 3.3, with inputs from [41, 48]. The process is carried out iteratively and repeatedly [41].

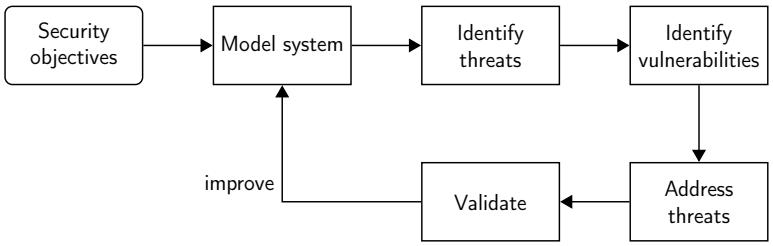


Figure 3.3: Threat modelling process

Threat modelling has a close relationship with risk assessment. Threat modelling is actually listed as "Threat Risk Modelling" in OWASP database [48]. We must emphasise that although risk assessment is part of the process, but our goal is to reveal potential security vulnerabilities. Since our architectural threat analysis does not focus on probabilistic risk analysis, we only utilise attack tree and STRIDE threat listing methods.

The main limitation with threat modelling methods is the fact that these methods are highly dependant on the design and characteristics of the system under analysis. There has been comprehensive work carried out to provide a unified and generalised "one size fits all" solution for different system designs. We will mention the important developments of this category here, but for the purpose of this report and our threat modelling, we will only elaborate *STRIDE* and *attack tree* methods. Just the fact that there are numerous variations and methods for threat modelling, is an undeniable proof for the dependency of the practice on system design and internals.

3.7.1 STRIDE threat listing

The STRIDE method deals with threats in a categorical fashion. Each letter in STRIDE stands for one category of threats such that, every existent threat can fall under one category. Each of these threats violates a security property of the system. We will elaborate these categories in the following paragraphs [41].

Spoofing (S) Spoofing is the false claim of being someone or something else. In that sense, any unique identifier of any kind can be spoofed, such as, a person's name, an IP address, a DNS name, etc. Spoofing violates *authentication* property.

Tampering (T) tampering involves unauthorised modification of the data, in any form or way. Tampering could be applied to static data, i.e. a file, as well as to dynamic data, i.e. data in memory, or data streams in a network. The violated property is *integrity*.

Repudiation (R) Repudiation is to avoid responsibility for actions by claiming you have not carried them out. Quite obviously, repudiation violates *non-repudiation* property.

Information disclosure (I) Information disclosure is to provide data access to non-authorised entities. Similar to tampering, both static and dynamic data can be the subject. *Confidentiality* is the property being violated here.

Denial-of-Service (D) Denial-of-Service (DoS) attacks involve heavy resource utilisation, with the intention of service disruption. It applies to both computational processing and communication resources. DoS violates *availability*.

Elevation of privilege (E) This category involves users and processes initiating actions they are not authorised to. The violated security property here is *authorisation*.

There are two major types of STRIDE, namely, STRIDE-per-Element and STRIDE-per-Interaction [41]. The latter supersedes the formers in terms of provided detail. Table 3.2 shows the applicability of threat categories on elements of a DFD for STRIDE-per-Element (taken from [41]).

Element	S	T	R	I	D	E
External entity	×		×			
Process	×	×	×	×	×	×
Data flow		×		×	×	
Data store		×	?	×	×	

Table 3.2: Threat category associations for STRIDE-per-Element (?: unknown, case based)

The more detailed STRIDE-per-Interaction flavour is included in Table 3.3 (taken from [41]). One can also expand these tables based on the system model at hand.

3.7.2 Attack trees

Attack trees are intended for software development processes and mostly involve analysis of virtual components. As far as the methodology and applicability of STRIDE is concerned, we can use attack trees in the same fashion for architectural diagrams as a replacement for STRIDE.

Bruce Schneier describes attack trees as "a formal way of describing the security of systems [35]." Attack trees can be used as building blocks for threat modelling and they are great as a threat-brainstorming tool to discover threats when analysing a system. Attack trees are also quite commonly created and thus, chances are, one can conveniently find an already developed attack tree for the same system, or another closely similar one [41].

Let us now take a look into the process of generating new attack trees from scratch. The steps are as follows [41],

1. decide on representation,

Element	Interaction	S	T	R	I	D	E
Process	Outbound data flow to data store	×			×		
	Outbound data flow to a process	×		×	×	×	×
	Outbound data flow to external interactor (code)	×		×	×	×	
	Outbound data flow to external interactor (human)			×			
	Inbound data flow from data store	×	×			×	×
	Inbound data flow from a process	×		×		×	×
	Inbound data flow from external interactor	×				×	×
Data flow	Crosses machine boundary		×		×	×	
Data store	Inbound data flow from process		×	×	×	×	
	Outbound data flow to process			×	×	×	
External interactor	Passes input to process	×		×	×		
	Gets input from process	×					

Table 3.3: Threat category associations for STRIDE-per-Interaction

2. create a root node,
3. create sub-nodes,
4. consider completeness,
5. prune the tree,
6. check the presentation.

The root node of an attack tree can be a problematic state or the overall goal of an attacker. Accordingly, sub-nodes will elaborate what can go wrong to achieve the goal of the root node. The representation can be either an OR tree, or an AND tree. One can also consider the combination of both, unnecessarily increasing the complexity of the attack tree. State of parent nodes in an AND tree depend on the truth value of all child nodes. Following the same pattern, tautology of parent nodes in an OR tree, requires only one child node to be true. Keep in mind that generating attack trees is not an exact science and there is room for personal or case related preferences [41].

The rest of the steps, completeness, pruning and the final check is all about efficiency of the presentation. In that sense, these steps result in inclusion of missing threats, removal of redundant threats and division of tree to more manageable sized sub-trees, respectively [41]. The most important goal is to have a human readable final result.

3.7.3 Attack-Defence Trees (ADTrees)

The next level of formalism based on attack trees is Attack-Defence Trees, or shortly, ADTrees [17]. An ADTree increases the complexity of an Attack Tree by allowing both AND type and OR

type branching. In other words, with ADTrees, we get both disjunctive and conjunctive refinements in our toolset. ADTree formalism also introduces the notions of *attack node* and *defence node*. Every node can have multiple children of the same type, but only one child of the opposite type. The intention behind this provision is to keep the tree readable and to the point. Connections between opposing modes are presented with a dotted line, representing a countermeasure, while the presentation of a conjunctive refinement involves an arc between the connectors of relevant child nodes [17].

3.7.4 CORAS

CORAS is a method and a graphical language for visualisation of security threats and risk analysis documentation [9]. Like any of the introduced methods, one can think of CORAS as a methodical brainstorming tool. CORAS is relatively new and one of the main motivations behind its conception is to generate understandable documentation and provide clear visualisations. People with different roles and backgrounds are involved in brainstorming and having such a tool will greatly facilitate the process [9].

CORAS language considers threats as part of the computerised system under scrutiny. On the other hand, vulnerabilities are seen as features of the same system. Needless to say, these are highly undesired features. Such an approach creates a direct link between CORAS and UML language, since both are designed to address features of a system. To cover the threat modelling elements, CORAS introduces three new diagram types under UML, namely, *asset diagrams*, *threat & unwanted incident diagrams* and *treatment diagrams* [53].

3.7.5 Boolean logic Driven Markov Processes (BDMP)

Boolean logic Driven Markov Processes (BDMP) is another modelling tool, combining fault trees with Markov processes, through defining Markov models for events. It is actually closer to a mathematical model than a threat model, but since it can be used to visualise risk, we have mentioned it here. BDMP is mostly used for dependability assessment and has two advantages in this regard, compared to other models. Firstly, complex dynamic models can be defined, while readability is almost the same as fault trees. Secondly, the mathematical basis of BDMP provides efficient processing capabilities [3]. A BDMP includes (taken from [3]),

- a multi-top coherent fault tree,
- a main top event,
- a set of triggers,
- a set of triggered Markov processes, associated with the basic events,
- the definition of two categories of states for the processes.

3.7.6 Bayesian Attack Graphs (BAG)

Yet another type of model focusing on risk assessment and mitigation is Bayesian Attack Graphs (BAG) [26]. BAG is particularly suitable for modelling network attacks and network systems. Poolsappasit et al. argue in their paper [26] that common models like attack graphs and attack

trees fall short. These models do not include causal dependencies between network states. Another benefit of BAG, according to the authors, is its attention to resource constraints of the system [26].

4 New terminologies

Author's introductions are described in this section, along with explanatory content and where it applies, examples from the literature.

4.1 Trust

The relation between security and trust is a clear one. Security processes, mechanisms and components involved in them must be trusted for those processes to be considered fulfilling their purpose. This requirement is even more important when the focus is privacy. Obviously, trust is not black and white and there can be different levels of it. You may trust a bank with your money, but you will trust close relations with your life.

To follow a methodical approach, we introduce the notions of *trusted secure* and *trusted private*. These two notions are generic and apply to any component involved in the process at hand. As an example, these components could apply to servers (services), databases (storage), client-side processes, communications channels, etc. Pay attention that available levels of trust has to be previously defined in the requirements of the system and anything below the accepted levels is simply not qualified. Thus, levels defined in one system does not simply apply to another. One can also sum up the levels of component to generate an overall trust level for the process including them. Alternatively, and this is the case for entities with obscured processes, such as third-party service providers, supported security enhancing and privacy preserving technologies can be used. Their combination will be an indicator for trust level of the system.

Trusted secure A trusted secure component is the one fulfilling a required level of security, based on the requirements specification. In the case of a subsystem, involving multiple components, the trust level is equal to the lowest trust level provided by any of the components. Hence, the weakest point of failure is the most detrimental for the system. Following from that, the trust level of the whole system is as good as the lowest trust level involved in its processes.

Trusted private A trusted private component, subsystem, or system is the one fulfilling a required level of privacy, based on the requirements specification. Similar to trusted secure, the level of trust for subsystems and systems is as good as the lowest trust level involved in their processes and subsystems, respectively.

It is a common practice to define different quantitative levels for qualities, based on their detailed characteristics and specifications. Consider Quality Authentication Assurance (QAA) framework [10] from the STORK project [47] as an example. STORK stands for Secure iden-Tity acrOss borDers linKed, aimed at facilitating the access to on-line services for public and private entities, within Europe. Different European countries have different identification and authentication methods. Therefore it is necessary to differentiate between the available method according to some indicator of their quality, credibility and reliability. In this sense, STORK

QAA defines four levels of assurance based on the involved risks for both registration phase and electronic authentication phase. These levels are numbered from 1 to 4, describing *no or minimal assurance*, *low assurance*, *substantial assurance* and *high assurance*, respectively [10]. Table 4.1, reproduced from [10], further elaborates the relation between registration phase, electronic authentication phase and different levels. An important observation is the exclusion of the quality of communication infrastructure in STORK QAA. For instance, resistance against Denial-of-Service (DoS) is beyond STORK's focus.

		Authentication Phase (AP) assurance levels			
		AP1	AP2	AP3	AP4
Registration Phase (RP) assurance levels	RP1	QAA level 1	QAA level 1	QAA level 1	QAA level 1
	RP2	QAA level 1	QAA level 2	QAA level 2	QAA level 2
	RP3	QAA level 1	QAA level 2	QAA level 3	QAA level 3
	RP4	QAA level 1	QAA level 2	QAA level 3	QAA level 4

Table 4.1: Different STORK QAA levels, resulting from registration phase and authentication phase levels

Organisations can come up with similar definitions for acceptable trust levels, based on their security requirements and privacy expectations. Such a table, with possibly more granular levels based on combinations of provided technologies as metrics, could act as a decision making tool for the choice of third-parties to be used with organisations. For instance, changing Registration Phase (RP) to Trusted Secure (TS) and Authentication Phase (AP) to Trusted Private (TP) in Table 4.1 provides such a template. From here onward in this report, *trusted* means conformity to predefined trusted secure and trusted private levels.

4.2 Bring Your Own Device (BYOD) technologies

Let us look into the access control possibilities with mobile devices, which can very well utilise biometric technology.

The notion of *Bring Your Own Device (BYOD)* is more of an umbrella term for different use-cases dealing with personal mobile devices in an organisation.These use-cases can vary from typical BYOD, where a member of the organisation brings their mobile device in the ICT premises and needs access to resources, while the ICT infrastructure must be kept secure in the process. It must be mentioned that, our view of BYOD is not from an information security perspective, which is the classical way of approaching BYOD. Instead, we look at BYOD as an element, or rather replacement, within identification and authentication processes of access control. With that in mind, when considering BYOD along with access control, there can be a more refined categorisation.

4.2.1 Bring Your Own Authenticator

Interpreting BYOD as Bring Your Own Authenticator would mean that the user will provide the information required for an authentication process to the access control system. By *authenticator* we mean what authenticates a user, e.g. passwords, biometric samples, or tokens [23].

An authenticator can either be a token (the final result of credential verification), or it can be raw credentials, used in validating the user. To differentiate between Bring Your Own Authenticator and Bring Your Own Authentication, it would make sense to abbreviate the former as *BYOAuthenticator (Bring Your Own Authenticator)*. On the other hand, a system accepting tokens will fall under the password/biometric sample-based category, since an initial verification is necessary for token generation. As a result, a token-based system with a process internal to the organisation is conceivable. Accordingly, the token will be kept on the mobile device and may be used as long as it is valid. Under BYOAuthenticator, users cannot choose the authentication process and must communicate their authenticator to the organisation. However, the organisation itself may use a trusted third-party for verification.

1. Template local to the mobile device: Template on device, sample collection on device, verification on internal server (may generate a token)

2. Template local to the organisation: Template on internal server, sample collection on device, verification on internal server (may generate a token)

4.2.2 Bring Your Own Authentication

In a *Bring Your Own Authentication (BYOAuthentication)* interpretation of BYOD, what we mean by *authentication* is the authenticating process, receiving user credentials and conducting verification. Here, the user is free to choose the authentication/verification process and uses the mobile device to access the relevant provider. Depending on the user's choice from third-parties trusted, or untrusted by the organisation requesting the authentication, there can be different cases. These third-party processes may or may not be local to the mobile device. For instance, a process internal to a smartphone, such as Apple's Touch ID, is a third-party authentication service.

1. Trusted authentication process via a local third-party, i.e. on the mobile device, meaning: Template on device, sample collection on device, verification on device (having a trusted authentication process implies that all participating elements are trusted)

2. Trusted authentication process via a remote third-party, i.e. AaaS on cloud, meaning: Template on remote server, sample collection on device, verification on remote server

3. Untrusted authentication process via a local third-party, i.e. on the mobile device, meaning: Template on device, sample collection on device, verification on device (having an untrusted authentication process implies that at least one participating element is untrusted)

4. Untrusted authentication process via a remote third-party, i.e. AaaS on cloud, meaning: Template on device, sample collection on device, verification on remote server

There may very well be use-cases for untrusted authentication processes, but our focus in this study is solely on trusted authentication processes.

5 System modelling for PACS

As demonstrated in Subsection 3.7, the first step in the threat modelling process is to model the system. Modelling a system includes two major steps, architectural modelling and deriving a DFD from the architecture. In this section, we conduct these processes step-by-step.

5.1 Generic PACS architecture

To have a better understanding of the interactions and the connectivity between components aforementioned in Subsection 3.1.2, as well as laying the foundation for threat modelling, we will introduce a generic architecture for PACS. This architecture, depicted in Figure 5.1, is not representative for every possible implementation, but it is one of the most common set-ups.

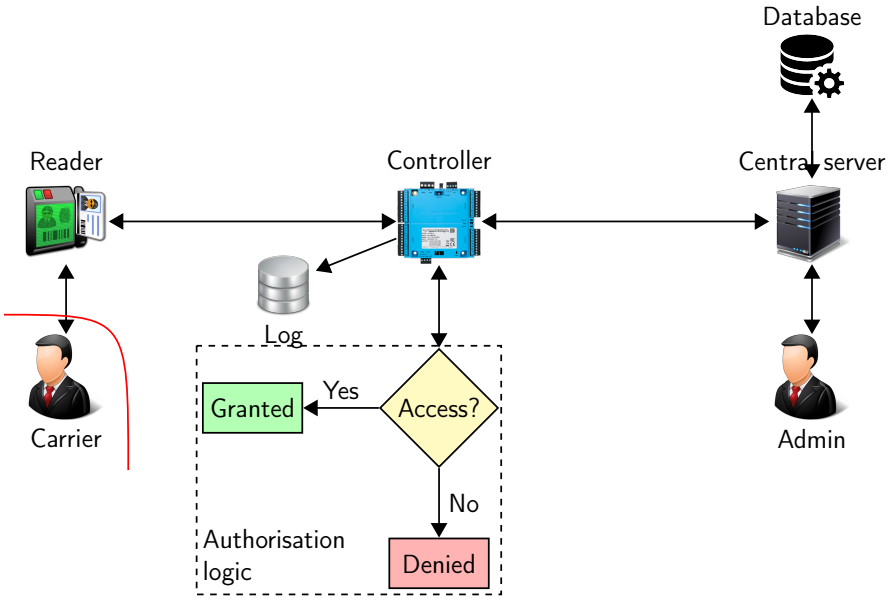


Figure 5.1: A generic PACS architecture, including authorisation functionality

This generic architecture can be paired with possible methods for identification such as, *biometric ID* and *card ID*, as well as authentication mechanisms such as, *biometric verification*, *certificate verification* and *PIN verification*. Biometric ID as an identification method is mainly used because of its provided convenience, compared to smart card, or PIN verification. In majority of cases, the industry does not combine biometric identification with any authentication mechanism. As mentioned, convenience and speed of transaction is the aim with biometric ID. We on the other hand, are looking for a combination of identification and authentication. Thus, we will assume card ID identification method, in combination with biometric authentication mechanism. Moreover, although we are assuming both non-biometric and biometric methods in our model through multipurpose readers, we will only focus on biometric authentication mechanism. Dependence of authorisation on identification is a known fact, but we will not depict identification in any of our diagrams and assume its proper execution.

5.2 Architectures involving biometrics

To keep the report focused and to abide by the fundamental security principles, architectures presented here may differ from the industrial best-practices. Our goal is to improve security and privacy, while the industry's foremost concern is convenience. Also, to make our depictions clear and avoid unnecessary clutter, we will consider purely biometrics-based designs from here on. This practice will not affect our results, since any fact derived from a purely biometrics-based design is valid for a mixed architecture as well.

It has to be mentioned that the biometric verifier can be thought of as a platform, running the matcher process, which in turn compares samples and templates. This process can either run on one of the internal components, such as central server, or can be provided by a third-party. As you can see in the architectures, we have considered different cases. Considering the fact that in a biometrics set-up, the biometric verifier as a platform for the matching logic, along with the template storage are the most important components [33], we can have several generic cases. So basically, we are braking down the generic PACS architecture from Figure 5.1 to different cases based on the location of biometric verifier and template storage. These cases are listed below and will be explained in separate sections.

- An *internal* biometric verifier and an *internal* template storage, depicted in Figure 5.2.
- An *internal* biometric verifier and an *external* template storage, depicted in Figure 5.3.
- An *external* biometric verifier and an *external* template storage, depicted in Figure 5.4.
- An *external* biometric verifier and an *internal* template storage, depicted in Figure 5.5.

Here, *internal* means components under the control of the organisation, while *external* means components are not under organisation's control. A third-party provider chosen by the organisation implies a very high level of trust and is considered same as an internal component, while a third-party provider chosen by externals is as an external component.

5.2.1 Internal biometric verifier and internal template storage

This case has both biometric verifier and template storage as internal to the PACS implementation. The two components are usually consolidated with the central server into a single unit, but this does not concern the ultimate goal of our analysis, which is threat modelling. A biometric sample is provided to the biometric verifier either directly by the biometric reader, or through the controller. It is important to mention the role of the *log* component here. Any system providing authentication and authorisation functionalities requires logging for auditing purposes. The interaction between the controller and the log is to record authorisation actions. The interaction between the biometric verifier and the log though, is to prevent hill climbing attack, which is a special case of brute-force attack. Obviously, the logging for authorisation and verification could be separate, but then again, this does not affect our analysis. Figure 5.2 depicts this architecture.

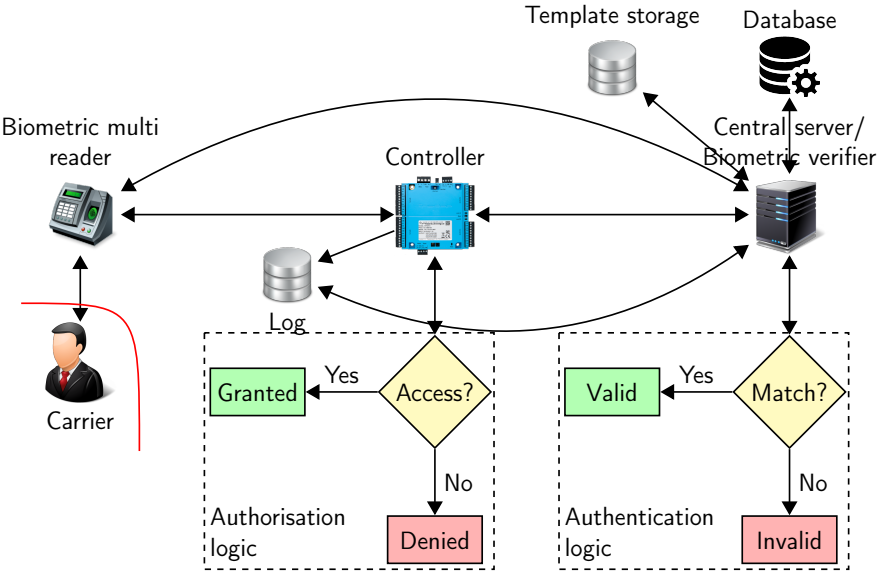


Figure 5.2: A PACS architecture with internal biometric verifier and internal template storage, including authentication and authorisation functionalities

5.2.2 Internal biometric verifier and external template storage

This case still utilises an internal biometric verifier, but the template storage is external. What we mean by external is that the component is not under our control. Here the example of a smart card has been used as the storage for the biometric template. Later on, we will mention that the smart card can be replaced by a mobile device. We can also consider an external third-party storage as the template storage. Figure 5.3 depicts this architecture.

5.2.3 External biometric verifier and external template storage

Another straight forward case is to outsource the process of biometric verification completely, including the template storage. This can simply be interpreted as using cloud-based biometric verification, along with integrated cloud-based storage. Such an implementation can be very appealing for organisations in terms of usability and convenience, but we must emphasise that in such a case, both biometric verifier and template storage will be outside the control of the organisation and external to the system. What makes the whole concept of utilising a third-party for biometrics verification, or template storage sensitive is the difference in trust level. Obviously, the trust in a third-party is much lower by default, unless it is guaranteed by security and privacy preserving technologies. Figure 5.4 depicts this architecture.

5.2.4 External biometric verifier and internal template storage

Lastly, the verification is provided by an external third-party, while the template storage is internal to the implementation. The motivation behind such a design can be the desire to be in full control of the template storage and the lack of biometric verification technology by the PACS. Although, one can argue that it is not necessarily an advantage to keep templates internal to the system, since these are communicated to the third-party for verification anyway. Figure 5.5 depicts this architecture.

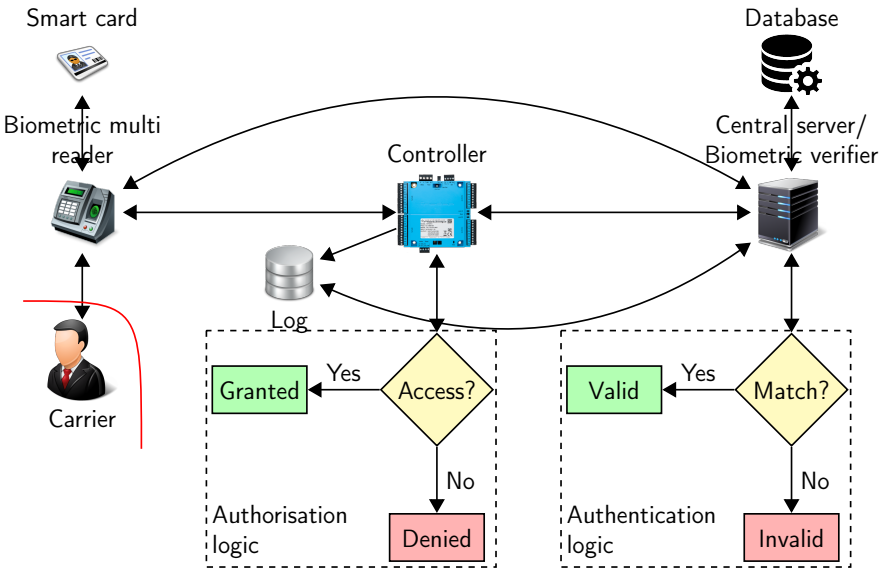


Figure 5.3: A PACS architecture with internal biometric verifier and external template storage, including authentication and authorisation functionalities

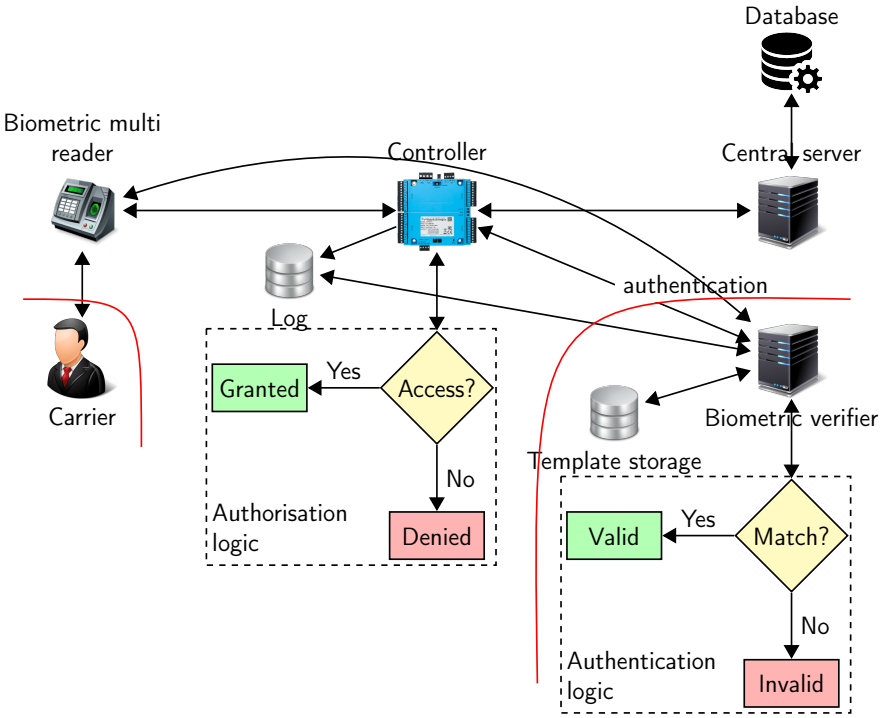


Figure 5.4: A PACS architecture with external biometric verifier and external template storage, including authentication and authorisation functionalities

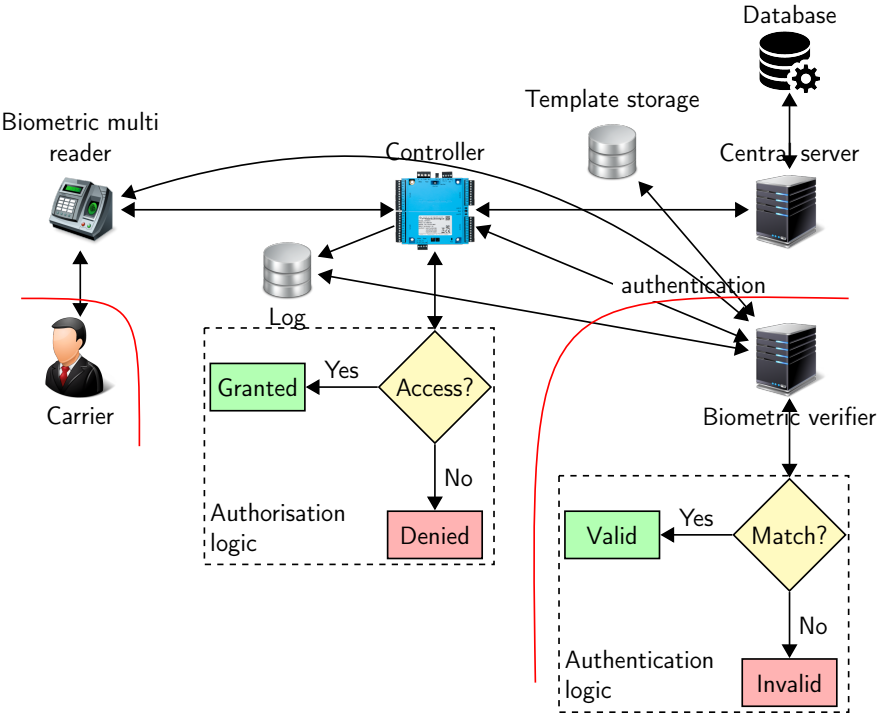


Figure 5.5: A PACS architecture with external biometric verifier and internal template storage, including authentication and authorisation functionalities

5.3 Architectures involving biometrics via mobile devices

The next batch of architectures introduced here, depict mobile devices in a PACS ecosystem. We will base these architectures on our definitions of BYOAuthentication and we will place them in the right category, from those aforementioned in Subsection 5.2.

5.3.1 Internal biometric verifier and internal template storage

The architecture given in Figure 5.6 closely follows the architecture from Subsection 5.2.1, with internal biometric verifier and internal template storage. The only difference here is the use of a mobile device, instead of a static biometric reader. Mind you, a mobile device, unlike a static reader, is considered external to the system. Therefore, sample collection is handled externally. In short, the architecture given here, corresponding to the second case of BYOAuthenticator, falls under *internal verifier-internal storage* PACS architecture.

5.3.2 Internal biometric verifier and external template storage

Similarly, the architecture depicted in Figure 5.7 correlates to the one explained in Subsection 5.3.2, with internal biometric verifier and external template storage. Here, the mobile device acts as the combination of a static reader and a smart card, replacing both. Pay attention that a mobile device is external to the organisation, while a smart card is considered internal, since it is issued by the organisation. In short, the architecture for the first case of BYOAuthenticator, falls under *internal verifier-external storage* PACS architecture.

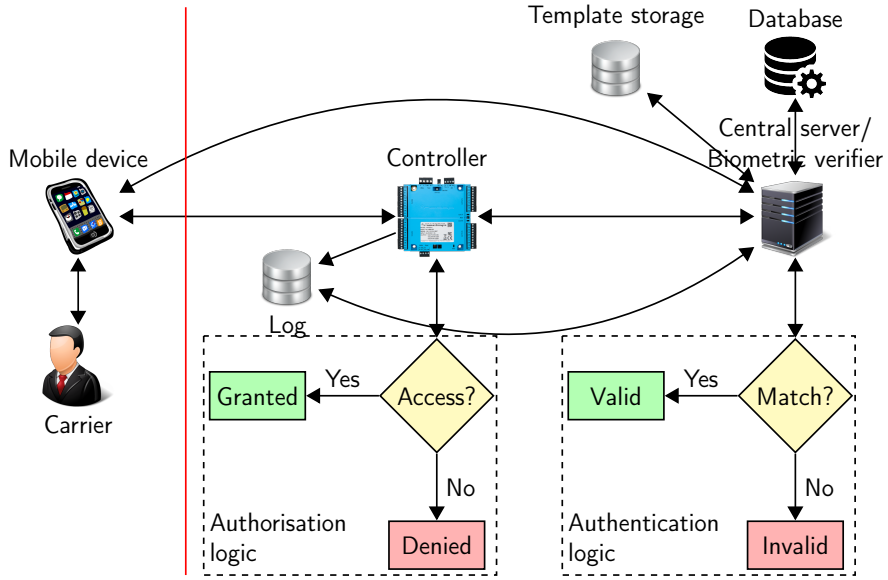


Figure 5.6: A PACS architecture with internal biometric verifier and internal template storage, involving a mobile device as BYOAuthenticator (internal authentication process)

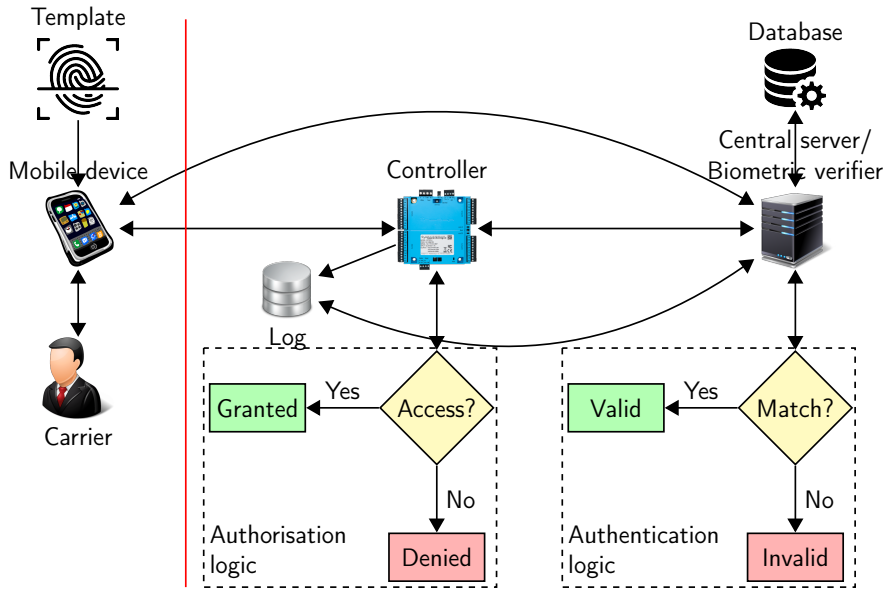


Figure 5.7: A PACS architecture with internal biometric verifier and external template storage, involving a mobile device as BYOAuthenticator (internal authentication process)

Note that in both cases of BYOAuthenticator, we have mentioned the possibility of using a trusted third-party for biometric verification. Although such a choice would mean having an *external verifier*, but since we consider such an outsourcing only to a *trusted* third-party, from our perspective, it will be the same as an *internal verifier*.

5.3.3 External biometric verifier and external template storage

Figures 5.8 and 5.9 both depict cases for BYOAuthentication, with the former being the case of a local third-party and the latter being the case of a remote third-party. Here, locality and remoteness are in relation with the mobile device and not the organisation. The authentication process in both cases is external to the organisation, hence they are categorised under this subsection.

Originally, we had four scenarios in Subsection 4.2.2, two for trusted authentication processes and two for untrusted. We can consider trusted and untrusted almost the same if the organisation has no say in the choice and gives the carrier complete liberty. However, the organisation might only accept trusted scenarios. In either way, there will be two cases as depicted in Figures 5.8 and 5.9.

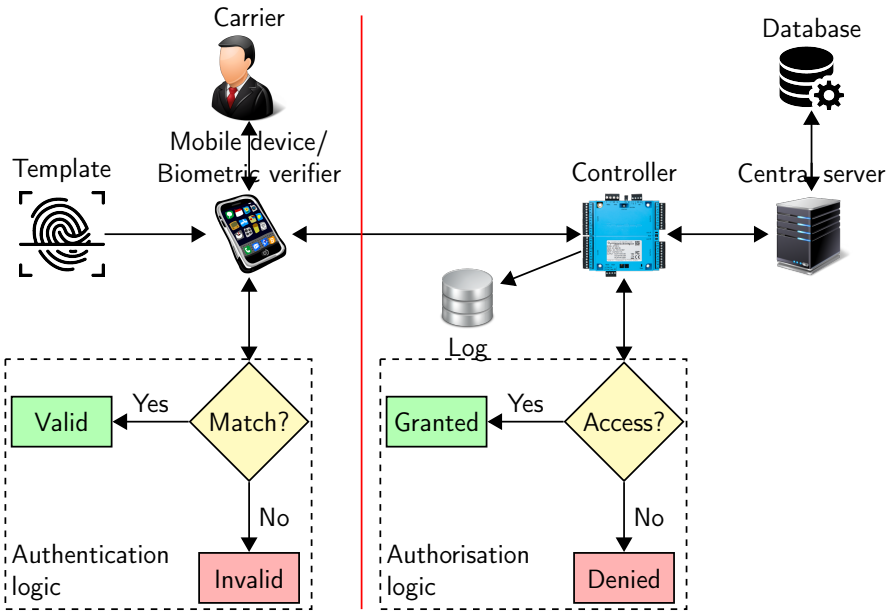


Figure 5.8: A PACS architecture with external biometric verifier and external template storage, involving a mobile device as BYOAuthentication (external authentication process)

5.3.4 External biometric verifier and internal template storage

As we have noted in Subsection 5.3.2, having an external biometric verifier and an internal template storage for a mobile device architecture can only be realised in the second case of BYOAuthenticator. This would require the organisation to use a trusted third-party biometric verification service. Since this third-party is trusted, from our point of view, the architecture and its following considerations are the same as Figure 5.6.

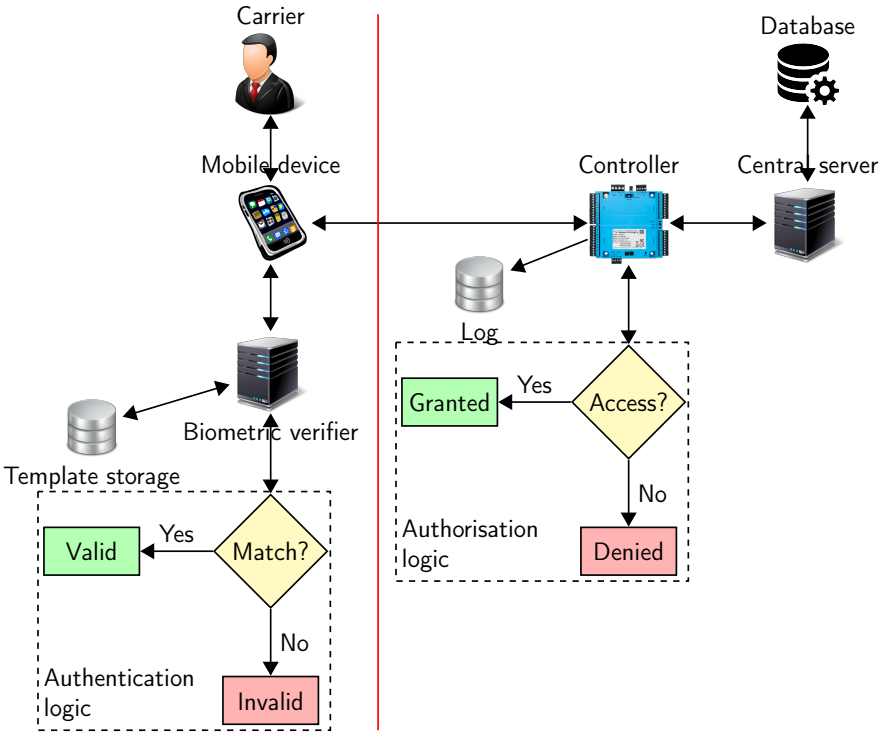


Figure 5.9: A PACS architecture with external biometric verifier and external template storage, involving a mobile device as BYOAuthentication and a third-party (external authentication process)

5.4 Data Flow Diagram (DFD)

As the first step in the threat modelling process, we need to generate a high-level diagram of the system. A diagram encompassing processes and modules of the system, along with their high-level interaction and messaging, would be an ideal candidate. The most suitable such diagram for threat modelling historically has been a Data Flow Diagram (DFD). The use of DFDs in threat modelling is so common that they are also referred to as *threat model diagrams* [41]. DFDs include the following elements,

- Process: A functional unit, running code, represented by a circle,
- Complex process: A complex functional unit, running code, represented by a double circle,
- Data store: A data storage unit, represented by two parallel lines around a label,
- External entity: Any real or virtual entity outside the system's control, represented by a rectangle,
- Data flow: Communication between any of these elements, which is two-way in most cases, represented by arrows,
- Privilege boundary: The boundary between different DFD elements, having different privilege levels, depicted as dashed lines. These differences depend on the elements and their

roles. Examples are, machine boundary (based on different data anonymity), integrity boundary (based on different trust level), process boundary (user vs. system), OS kernel mode vs. user mode, etc. [42],

- External boundary: We introduce this element as the separation line between internal and external entities, depicted as solid lines. This is especially necessary for clarity of our DFDs, since we differentiate between internal and external elements.

The amount of detail in a DFD is up to the modeller. There can be DFD levels such as, context diagram, level 0 diagram, level 1 diagram, etc., increasing the amount of detail as levels progress [42]. Modellers can stop elaboration as soon as they are happy with the amount of detail and see the diagram fit for capturing threats. DFDs are quite flexible and can be utilised to model a single software, or an operational network. We will see how this flexibility will serve us when modelling a network of interconnecting software for our threat modelling.

5.4.1 DFDs for PACS

Let us generate the DFD corresponding to the first PACS architecture with biometrics, Figure 5.2. To make the diagram more straight forward and avoid clutter, we only include components with direct influence in authentication processes. The result will be the depiction in Figure 5.10. Pay attention that in this particular case, the biometrics verifier is internal to the organisation. Otherwise, it should have been drawn as an external entity, using the rectangle shape.

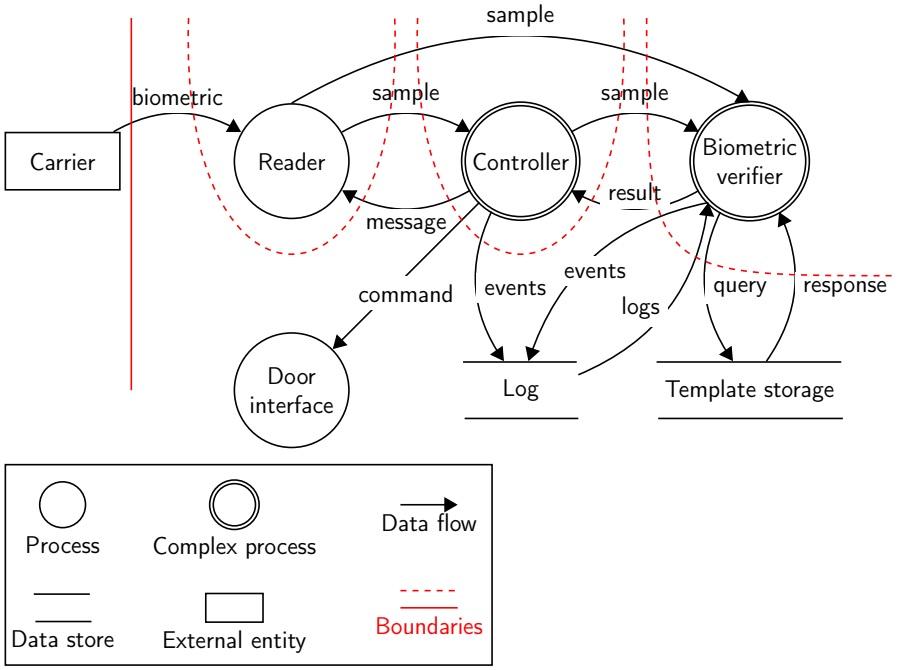


Figure 5.10: Data flow diagram for a PACS architecture

The case of an external biometric verifier, along with an external template storage, based on Figure 5.4 is depicted in Figure 5.11. As we mentioned, here the biometric verifier is external, hence the rectangle shape.

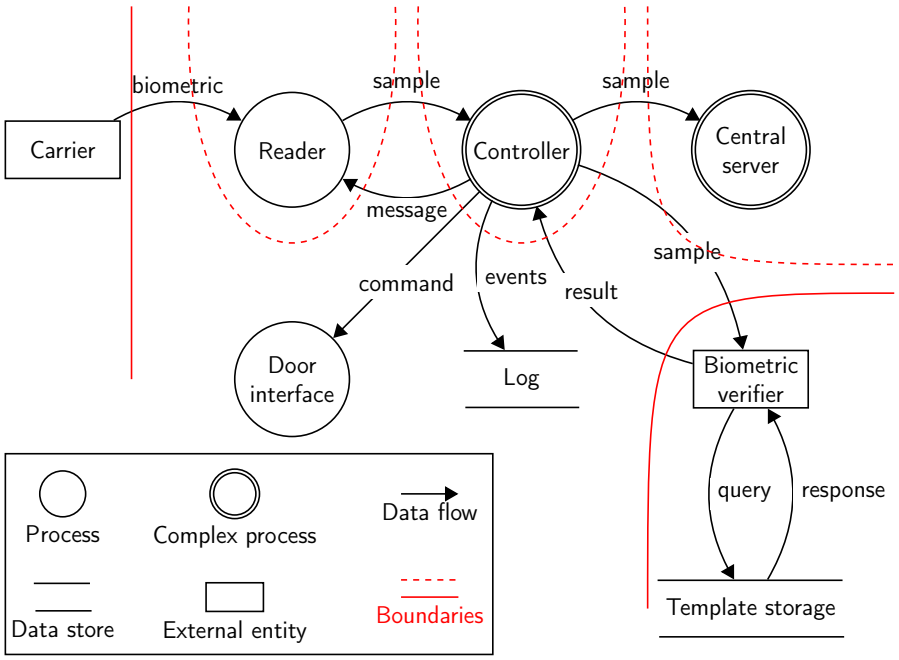


Figure 5.11: Data flow diagram for a PACS architecture, using third-party biometric verifiers

5.4.2 DFDs for PACS involving mobile devices

When it comes to DFDs for architectures of PACS involving mobile devices, two DFDs covering most of the change can be considered. The DFD for an internal biometric verifier, along with a template on the mobile device can be seen in Figure 5.12. In this depiction, generated from the architecture given in Figure 5.7 and corresponding to the first case of BYOAuthenticator from Subsection 4.2.1, differences such as the replacement of external boundary, are noticeable. Note that the case of internal biometric verifier and internal template storage, depicted in Figure 5.13 will be more or less the same as Figure 5.10, with changes such as external boundary and the mobile device replacing the reader.

The other important DFD for a PACS involving mobile devices is the case of an external biometric verifier. Considering a third-party verifier with a template storage, the resulting DFD would be of the form depicted in Figure 5.14. The changes here are more substantial. The whole verification process external now and the logging capability related to verification is also non-existent from the organisation's perspective.

5.5 General security and privacy concerns

Considering the introduced architectures and the included components, as well as communication links between these components, one can quickly brainstorm important security and privacy concerns. This could involve storage, data transmission, processes, hardware, etc. To follow a methodical approach however, we will conduct a threat modelling in Section 6, based on biometric elements.

Considering standard biometric elements, arguably, the most sensitive pieces of data in a PACS involving biometrics are biometric templates and biometric samples. These are unique biological features and as a result, unique identifiers of the owner. Unlike knowledge-based

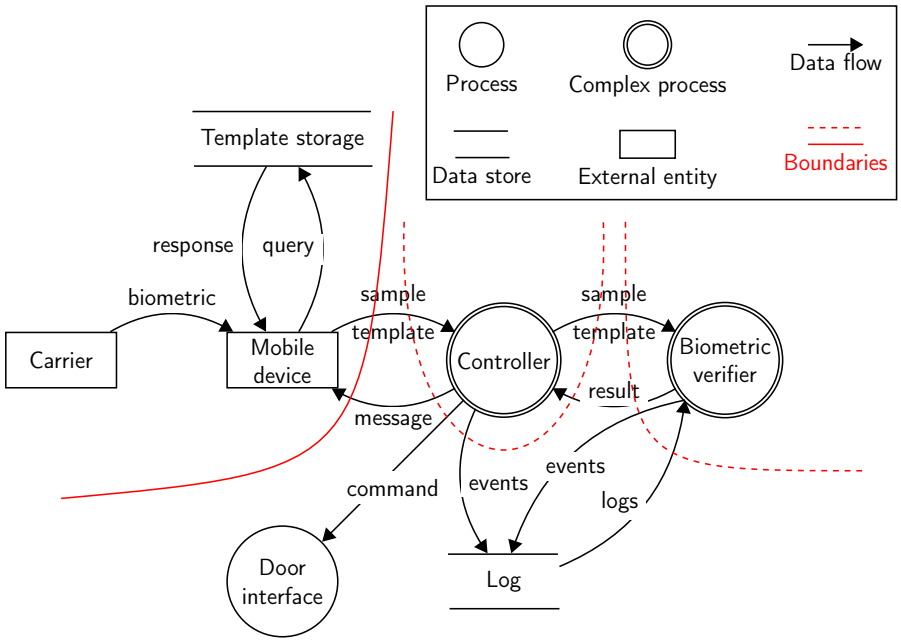


Figure 5.12: Data flow diagram for a PACS architecture, involving mobile devices as biometric readers and internal verifier

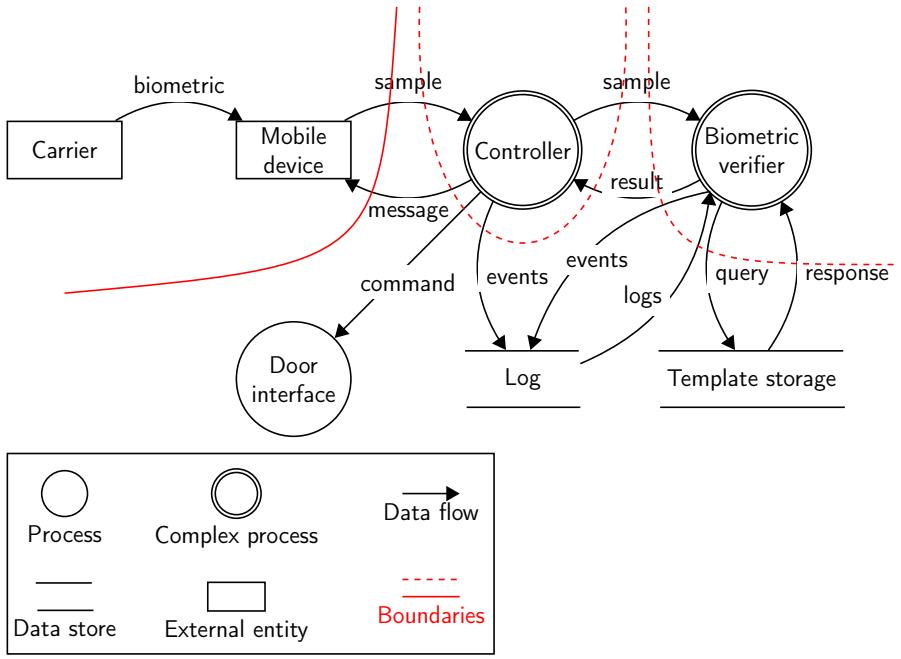


Figure 5.13: Data flow diagram for a PACS architecture, involving mobile devices as biometric readers and internal verifier with template storage

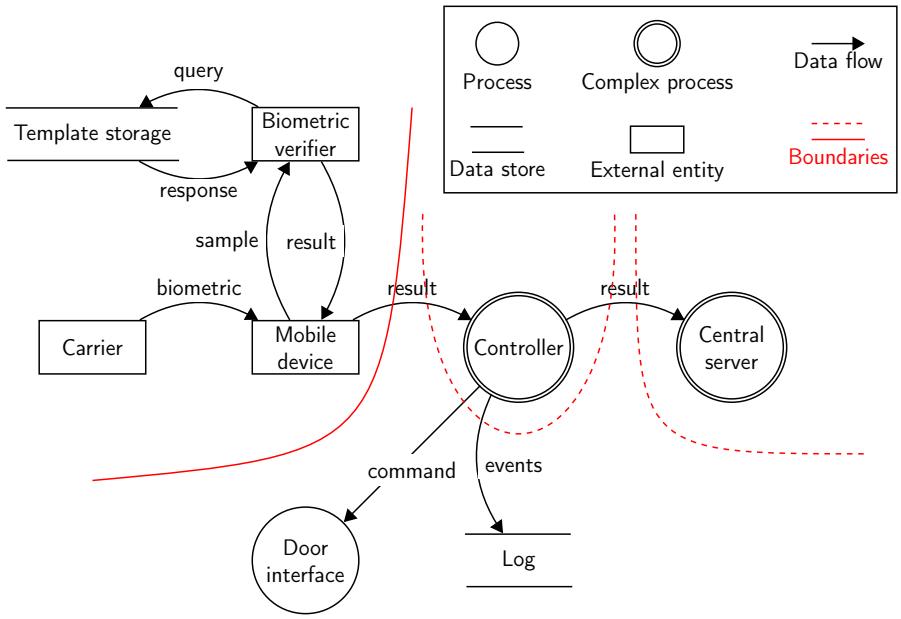


Figure 5.14: Data flow diagram for a PACS architecture, involving mobile devices as biometric readers and external verifier

methods, such as passwords, a compromised unique biometric identifier will be forever compromised. Therefore, threats and relevant mitigation strategies are highly influenced by the placement of these pieces of data, as well as the placement of the biometric verifier.

6 Attacks on PACS

The ultimate goal when attacking a PACS is to gain access to the protected physical site via authorisation. Based on how a PACS works and our predefined constraints, identification using a smart card and verification through biometric attributes, a number of steps are required. It is fair to assume that if someone has a valid ID, they also can authenticate, otherwise the ID would be non-existent, or suspended. Therefore, a malicious perpetrator must get identified, get authenticated and also have necessary privileges. The attack tree presented in Figure 6.2 includes these branches and elaborates the "get authenticated" branch, which involves biometric specific attacks.

6.1 Threat agents

As defined in [6], a threat agent is an entity, capable of conducting adverse actions over system, or system artefacts. There can be three groups of threat agents for our architectures.

Administrator Administrator refers to privileged users with access to IT infrastructure and resources. These can be internal to the organisation, or internal to involved third-parties. These entities have higher privileges compared to other users of the enterprise services and resources. This is a necessary factor, allowing them to perform administrative tasks, but that does not indicate any level of trust, or mistrust. An administrator with no malicious intent could still

perform an adverse action through a mistake. A threat analysis should consider the potential role of internal threat agents, as well as external ones.

Carrier Carriers are users, identified to the system, meaning they have some form of identification to provide and a corresponding record with the system, e.g. a smartcard, a certificate, or a PIN. This implies that they also have a biometric template with the system (not necessarily with the organisation), searchable using their identity. We do not care about the authorisation privileges of users, since we only focus on authentication. Although we can assume, if a carrier is performing an adverse action, it is aimed to gain authorisation. Another important factor regarding a *carrier* is that, under scenarios involving mobile devices, they do have legitimate access to the mobile device.

Attacker An attacker is an unidentified external entity who cannot and should not be identified, authenticated and authorised. To be more clear, we consider the attacker to be an animate, intelligent entity with solely malicious intent, aiming to perform adverse actions. Another important factor regarding an *attacker* is that, under scenarios involving mobile devices, they do not have legitimate access to the mobile device.

6.2 Artefacts

Any type of data, directly involved in the authentication workflow is considered an artefact. By artefact, we only mean and focus on security related artefacts and not the business related ones. Rhee et al. [31] use the term asset instead, but consider business valuables as well, which is not the case here. For a biometrics-based authentication system, we divide artefacts into three categories.

Temporary artefacts (deliverables) Deliverables are artefacts that are temporary to the system during an authentication workflow and need to be regenerated for every attempt. In that sense we will have biometric attribute, sensor raw data and biometric sample as temporary artefacts.

Persistent artefacts Persistent artefacts are unchanging for all authentication workflows. These are either stored in the system, or have equal regenerations throughout any number of authentications. Examples are biometric template and template transmission.

Result We consider the result as a separate category since it is neither an input to the authentication workflow, nor is processed during the authentication workflow, but it is the outcome of it.

6.3 Biometrics specific attacks

When it comes to attack vector listing from the perspective of biometrics, there are publications to start with. Ratha et al. [28] have given a detailed look into biometric authentication systems and unique security and privacy problems associated with them. They argue that there are eight possible attack vectors in a generic biometrics-based authentication system, involving

biometric authentication. Zamboni [56] improves on this by adding a ninth attack vector, including the template generation during the enrolment workflow as well. Mind you, attack vectors 1-3 still apply for an enrolment workflow. These nine points are depicted over a generic system representation in Figure 6.1.

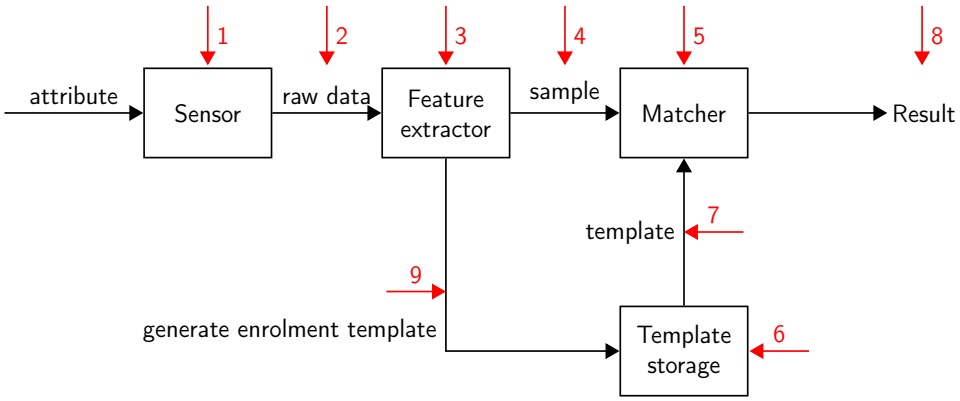


Figure 6.1: Attack vectors in a generic biometrics-based authentication system

The following enumeration is a short descriptive list of these attack vectors [28, 56]. Some of these attacks might seem infeasible, unless the modules of the system are remote to each other and are communicating over long distance connections, e.g. over the Internet, or an intranet. It can be clearly seen which STRIDE category these attacks belong to.

1. Attacking through the sensor: The sensor will be presented with fake biometrics, resembling the original.

2. Attacking through the raw data transmission link: Like any replay attack, raw data from a legitimate biometric sample is recorded and sent again later.

3. Attacking through the feature extractor: The process itself will be altered to generate malicious entity's desired sample.

4. Attacking through the biometric sample transmission link: Biometric sample transmission is tampered with before matching, using another fraudulent one. Replay can also be done.

5. Attacking through the matcher: The process itself will be altered to generate malicious entity's desired matching result and score.

6. Attacking through the biometric template storage: Any form of template storage, e.g. smart card, local, or remote, is susceptible to tampering. This could result in allowing a fraudulent entity, or denying a legitimate one.

7. Attacking through the biometric template transmission link: Biometric template transmission is tampered with, using another fraudulent one before matching. Capturing can also be done.

8. Attacking through the final result in transmission: The result is tampered with and will be changed to malicious entity's desired one.

9. Attacking through the enrolment template transmission link: Enrolment template transmission is tampered with, using another fraudulent one before storing. Capturing can also be done.

Although Roberts [33] argues that such frameworks are too complex and practical approaches are more to the point, we must reiterate that such categorisations are a necessity. Without having such attack taxonomies, a practical view would be an elaborate guess work, at best.

Following this list of attack vectors, we can immediately divide them in three major categories, namely, spoofing temporary artefacts, tampering with processes/transmission and gaining access to template storage. There can be other less relevant (from our perspective) categories, such as, denial-of-service. In such a case, an attacker will prevent a legitimate user from gaining access. As an honourable mention, we would like to emphasise on the importance of logging and its protection for a system, while integrating biometrics. Biometrics specific attacks, such as the hill-climbing attacks [43, 51], work on the basis of iterative improvement of trials. These actions would create a large number of undesirable logs about the attacker's activity. Removal of these logs is the first and foremost step towards repudiation. The attack tree depicted in Figure 6.2 visualises these details for biometric verification workflow. Therefore, we have not included attack vector 9 here. We are using logic gates, similar to fault trees, for legibility of conjunctions and disjunctions in our attack trees.

One of the practices when drawing an attack tree is to break the tree into subtrees as soon as legibility was being affected negatively, due to the increase in tree size. Accordingly, we expand subtrees in Figure 6.3 by including major steps needed for attack vectors. As an example, a hill-climbing attack is a variation of the attack vector 4, "spoof sample", and will fall under brute-forcing the sample, although, it is more efficient compared to a plain brute-force. This is the case where consistency of log is of importance.

A good way of checking the correctness of attack trees, at least in our case, is to cross-check the structure with notions of *target* and *enabler* [25]. These are resources utilised by attackers. Simply put, a target is the immediate goal of an attack vector, resulting in the compromise of the system. Enablers are the elements involved in achieving the target. In that sense, for us, different artefacts are targets, e.g. template, sample and result. The same way, enablers are elements from DFD that are involved in attack vectors, e.g. template storage, mobile device, controller, etc.

Let us now elaborate the main attack categories, which are spoofing attempts focusing on different targets, temporary and persistent artefacts, as well as the result.

6.3.1 Spoofing temporary artefacts

What we mean by temporary artefacts here are those pieces of information that are temporary to the system. This category of spoofing needs to be conducted every time a malicious user wishes to gain access to resources. The spoofing category applies to actual biometric attributes themselves, biometric raw data sent to the feature extractor and the digitally processed form of the biometric attribute, which is the sample. A perpetrator can potentially copy and reproduce the biometric attribute itself (Figure 6.3a). It is also possible to capture and replay the raw data sent to the feature extractor (Figure 6.3b). The perpetrator can also tamper with the feature extractor to change the result of the process, to a legitimate sample (Figure 6.3c).

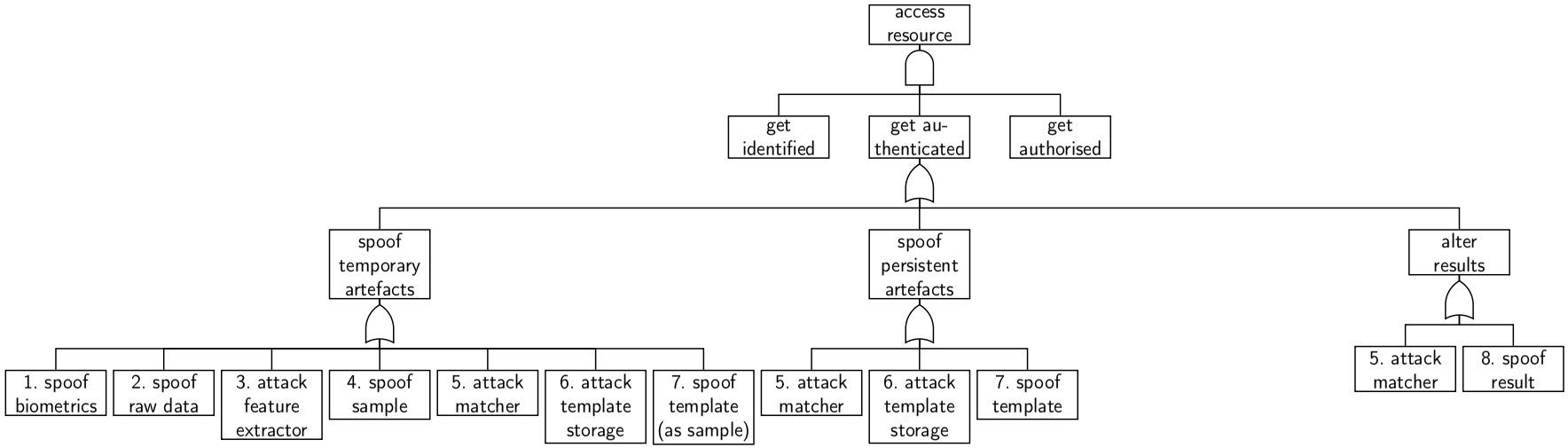
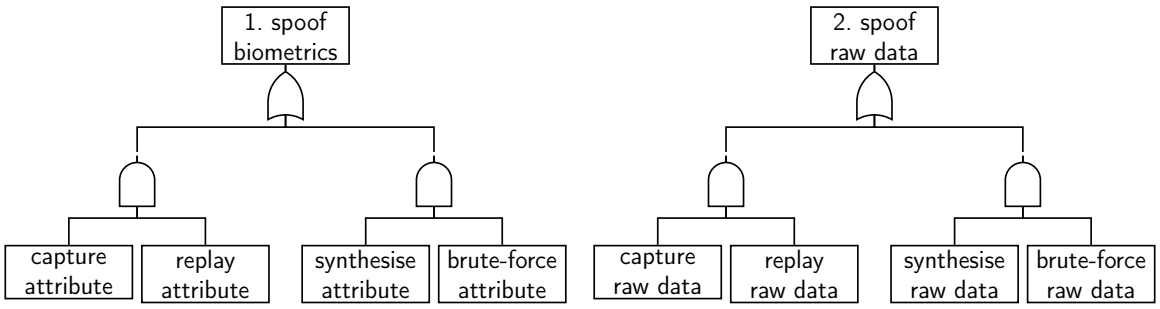
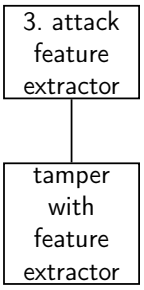


Figure 6.2: The attack tree for biometric verification workflow (AND gate: all child nodes must be realised, OR gate: at least one child node must be realised)

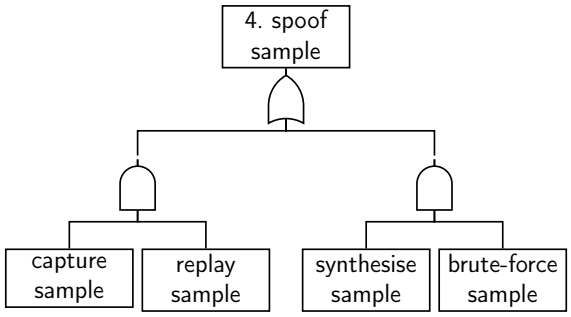


(a) Attack vector 1 tree

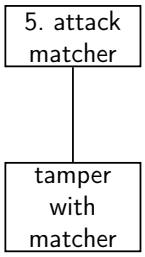
(b) Attack vector 2 tree



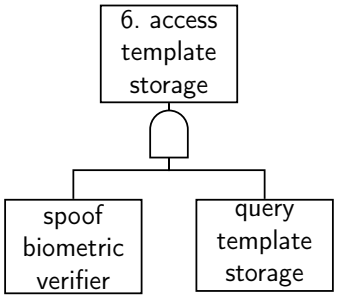
(c) Attack vector 3 tree



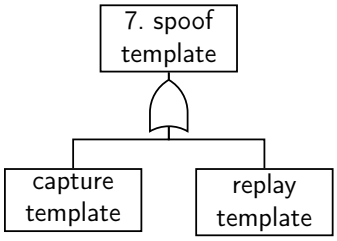
(d) Attack vector 4 tree



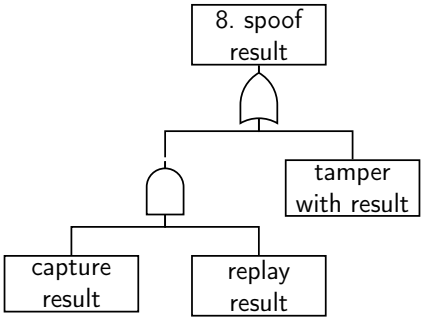
(e) Attack vector 5 tree



(f) Attack vector 6 tree



(g) Attack vector 7 tree



(h) Attack vector 8 tree

Figure 6.3: Detailed attack subtrees for each attack vector (leaf) in Figure 6.2 (AND gate: all child nodes must be realised, OR gate: at least one child node must be realised)

The sample itself can be sniffed too, while being transmitted to the matcher and replayed later on to gain access (Figure 6.3d). Finally, as another temporary form of data, a template in transmission from template storage to matcher can be sniffed and replayed (Figure 6.3g). Generally speaking, a template can be used as a sample, so any form of access to templates can be in this category of attacks.

6.3.2 Spoofing persistent artefacts

Spoofing persistent artefacts, namely, templates, has a much more serious impact and higher risk involved with it. In this fashion, a malicious user can go through the standard processes and get authenticated. This type of attack only needs to succeed once. Spoofing persistent elements is done by planting them in the system. This could happen by the template storage being compromised in one way or another (Figure 6.3f). Another possibility is to tamper with, or replace the template being transmitted to the template storage during the enrolment phase. We have not included this case (attack vector 9) in the attack tree representation, since it only focuses on the verification workflow.

If we consider the matcher process as a persistent element of the system, it is also feasible to categorise an attack against it here. The matcher can be compromised in a way to always verify the sample of a malicious user, regardless of the template (Figure 6.3e).

6.4 Denial-of-service perspective

DoS aspects of tampering with the system is also quite important, as it would interrupt business processes and cost the organisation. Here, the goal is to prevent a legitimate user from gaining access to a physical resource. An attack tree focusing on DoS through biometrics-based authentication, would be quite similar to the one in Figure 6.2. The main difference to specifically mention here is that, instead of *get identified AND get authenticated AND get authorised*, we will have *deny identification OR deny authentication OR deny authorisation*. Denying any of these branches will result in denied access, hence *OR*. The tree focusing on the authentication branch can be observed in Figure 6.4.

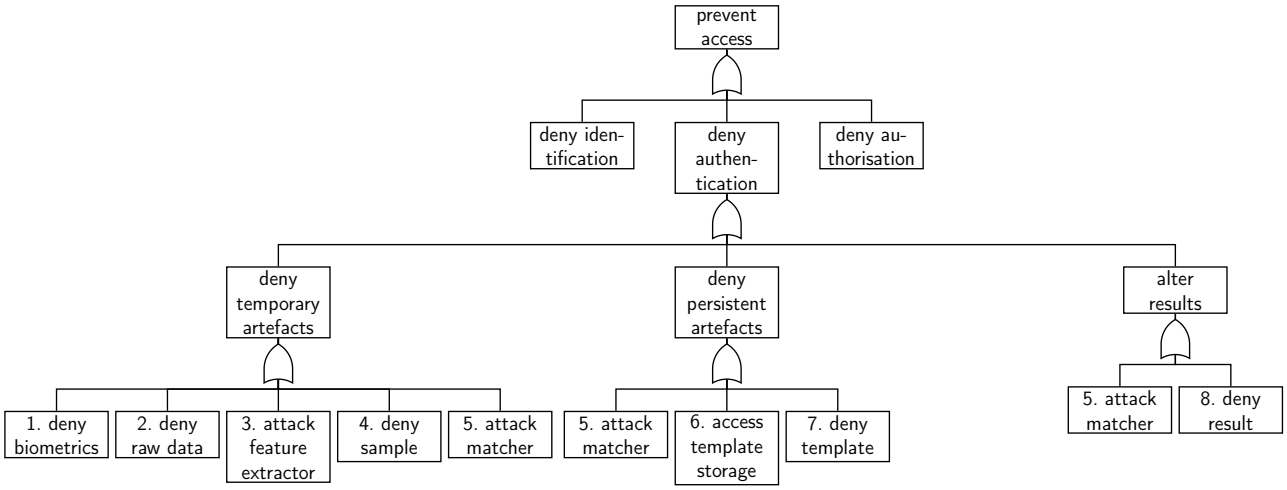


Figure 6.4: The attack tree for biometric verification workflow, focusing on DoS (OR gate: at least one child node must be realised)

6.5 Other cases

We have mentioned the importance of logging for prevention of biometric hill-climbing attacks. Logging can also be important for PACS auditing in certain scenarios. For instance, consider a legitimate internal collaborator, accessing physical resources on behalf of a malicious external entity. In such a case, the malicious external entity will be strongly interested in altering logs to cover the tracks of their internal collaborator. This case creates a separate attack tree, depicted in Figure 6.5, targeting the logging system.

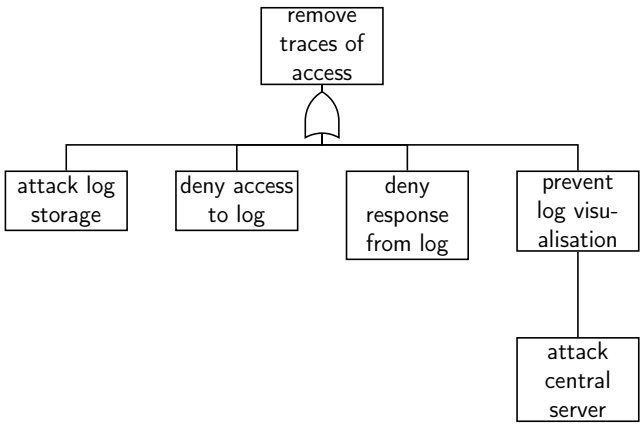


Figure 6.5: Attack tree for removing the traces of a legitimate collaborator (OR gate: at least one child node must be realised)

6.6 Threats against BYOD-PACS-biometrics

What we are presenting in this subsection is the culmination of this threat analysis and this is what the process boils down to. We are predominantly focusing on four scenarios involving mobile devices. Let us repeat these BYOAuthenticator and BYOAuthentication scenarios from Subsections 4.2.1 and 4.2.2 in detail here. We are omitting the ones involving untrusted third-parties, as such cases can be dismissed right away. The reason we introduced them in the first place was that there can be use-cases indifferent to security of the process.

Scenario 1 Biometric template is on the mobile device, biometric sample collection happens on the mobile device, while verification is done on an internal biometric verifier. The corresponding DFD is Figure 5.12. This is a case of BYOAuthenticator, with the authentication process being local to the organisation.

Scenario 2 Biometric template is on an internal template storage, biometric sample collection happens on the mobile device, while verification is done on an internal biometric verifier. The corresponding DFD is Figure 5.13. This is a case of BYOAuthenticator, with the authentication process being local to the organisation.

Scenario 3 Biometric template is on the mobile device, biometric sample collection happens on the mobile device, while verification is also done on the mobile device. The corresponding

DFD is Figure 5.14. This is a case of BYOAuthentication, with the authentication process being external to the organisation.

Scenario 4 Biometric template is on a remote third-party template storage, biometric sample collection happens on the mobile device, while verification is done on a remote third-party biometric verifier. The corresponding DFD is Figure 5.14. This is a case of BYOAuthentication, with the authentication process being external to the organisation.

The listing presented in Table 6.1 includes possible threats, along with relevant metadata presented in different columns, including,

- threat: reference to threat description, including sub-cases if different metadata in different scenarios are present,
- vector: attack vector from the attack tree,
- element: element included in the threat from DFD diagrams,
- agent: the agent performing the threat,
- artefact: targeted artefact in the threat,
- primary STRIDE ($*$): STRIDE categories the threat belongs to, referring to the initial action,
- secondary STRIDE ($\times$): STRIDE categories the threat belongs to, referring to later actions,
- scenario(s): scenario, or scenarios the threat could be applied to, and
- DFD(s): relevant data flow diagram(s).

While an attack vector describes *where*, a threat describes *how*. With that in mind, let us elaborate the actual descriptions of these potential threats. It must be reiterated that these are high-level descriptions and further threat details, which is out of our scope, would require more fine-grained architecture/DFDs, as well as information about the specific technologies and protocols. Also, as a result of high-level threat descriptions, each threat involves different STRIDE categories at the same time. This does not mean that the threat can be done in different ways, but instead, all marked STRIDE categories are different steps of the same threat. The most important one, which is usually the initial action, is distinguished by an asterisk mark as *primary STRIDE*.

Threat 1 Capture and reproduce fingerprints using putty and gelatin. Alternatively use fingerprint dust, or high resolution imaging and print them on paper or circuit boards. Wood glue is also another option. The result can be replayed to sensor. Capture a picture of a person and present it to the camera of a facial recognition system. This is applicable for both static and mobile sensors (cameras), but the success rate can be higher with mobile sensors as they are of lower quality. The mobile nature of mobile devices provides threat agents with easier, long-term access.

Threat 2 Brute-force fingerprints, or images by presenting the system with large databases of samples. Depending on technical details and the type of biometrics (fingerprint, facial recognition, etc.), brute-forcing may be done efficiently at sensor. Although this technique is a common test method for biometric algorithms in lab environments, but it is hard to perform in real-world. This is due to the unified structure of sensors and feature extractors, even at hardware-level. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much more flexible time constraints. Efficient brute-forcing would necessitate access to sensor's input feed, requiring tampering.

Threat 3 Capture the raw data transmission of a legitimate user from sensor to feature extractor and replay it later on. This also requires the relevant identification process result. This attack is hard to perform in real-world, due to the unified structure of sensors and feature extractors, even at hardware-level. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much more flexible time constraints.

Threat 4 Brute-force by generating random, or structured raw data and injecting them in the transmission from sensor to feature extractor. This can also be done using raw data generated from a database of samples. This attack is hard to perform in real-world, due to the unified structure of sensors and feature extractors, even at hardware-level. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much more flexible time constraints.

Threat 5 Tamper with feature extractor process (software), or algorithm, to create a sample output matching a legitimate user's template. This also requires the relevant identification process result. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much more flexible time constraints.

Threat 6 Tamper with feature extractor process (software) to capture a legitimate sample and replay it later on. This attack focuses on IO (Input/Output) functions of the process. Depending on software design, IO functions may or may not be easier to tamper with. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much more flexible time constraints.

Threat 7 Capture sample transmission of a legitimate user from feature extractor to matcher and replay it later on. This also requires the relevant identification process result. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much easier physical constraints, since the data flow has to use some type of wireless technology, possibly including intermediate communication devices. In such a case, the data flow itself is a complex element (not to be mistaken with complex process) and can be broken down for more granular DFD levels. Based on the location of the matcher, different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 8 Brute-force by generating samples in a hill-climbing attack and injecting them in the transmission from feature extractor to matcher. A hill-climbing attack uses the evaluation

score from the matcher to improve the transmitted sample and therefore is dependant on having access to it. This is the case for both static and mobile readers. Based on the given response from the matcher to the reader, having physical access to the matcher may, or may not be necessary. In case physical access is required, the attack is greatly facilitated in scenario 3. We do not cover the simple brute-force of a sample, since a hill-climbing attack is superior to it. For this attack we assume that the matching score is transmitted to the element handling authorisation (controller) and not the reader. A message informing the user of negative result does not include the score. Based on the location of the matcher, different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 8*[1] In addition to what is mentioned in threat 8 and considering the importance of logging in preventing hill-climbing attacks, attack the logging mechanism. This can be done either be tampering with the process sending events, by tampering with the logged events, by tampering with the transmission of events from biometric verifier to log data store, or by denying access to log data store. This is the case for both static and mobile readers.

Threat 9 Tamper with matcher process (software), or algorithm, to create a success result output. This also requires the relevant identification process result. This is the case for both static and mobile readers. The attack is greatly facilitated in scenario 3, where the matcher is on the mobile device and physically accessible.

Threat 10 Tamper with matcher process (software) to capture a legitimate sample and replay it later on. This attack focuses on network communication IO functions of the process. Depending on software design, network communication IO functions may or may not be easier to tamper with. This is the case for both static and mobile readers. The attack is greatly facilitated in scenario 3, where the matcher is on the mobile device and physically accessible. Based on the location of the matcher, different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 11 Tamper with matcher process (software) to send a query to the template storage, or capture a response from the template storage, and gain access to a legitimate template. The template can be replayed later on. This attack focuses on database (template storage) IO functions of the process. Depending on the software design, database IO functions may or may not be easier to tamper with. This is the case for both static and mobile readers. The attack is greatly facilitated in scenario 3, where the matcher is on the mobile device and physically accessible. Based on the location of the matcher, different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 12 Access template storage (database) directly to capture a legitimate template and replay it later on. This is the case for both static and mobile readers. The attack is greatly facilitated in scenarios 1 and 3, where the template storage is on the mobile device and physically accessible. The attack has a much larger potential impact in scenarios 2 and 4, where the template storage includes all user templates. Based on the location of the template

[1] This is an improved version of threat 8 and is not included in the extended STRIDE list.

storage, different threat agents have advantage to perform the attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 13 Capture query transmission of a legitimate user from matcher to template storage and replay it later on to receive the template. The template itself can be replayed afterwards. This is the case for both static and mobile readers. The attack is greatly facilitated in scenarios 1 and 3, where the template storage is on the mobile device and physically accessible. Based on the location of the template storage, different threat agents have advantage to perform the attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 14 Capture response transmission of a legitimate user from template storage to matcher and receive the template. The template can be replayed afterwards. This is the case for both static and mobile readers. The attack is greatly facilitated in scenarios 1 and 3, where the template storage is on the mobile device and physically accessible. Based on the location of the template storage, different threat agents have advantage to perform the attack. The risk is higher in scenario 1, since the response transmission travels through more elements and passes through the external boundary. This means that two different threat agents have advantage to perform the attack in scenario 1. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 15 Capture the result transmission of a successful match from matcher (biometric verifier) to the element handling authorisation (controller) and replay it later on. This may or may not require the relevant identification process result. This is the case for both static and mobile readers. The attack is greatly facilitated in scenarios 3 and 4, where the matcher is external to the organisation and the transmission is physically accessible to more threat agents. Different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Threat 16 Tamper with result transmission from matcher to controller and generate a success result. This may or may not require the relevant identification process result. This is the case for both static and mobile readers. The attack is greatly facilitated in scenarios 3 and 4, where the matcher is external to the organisation and the transmission is physically accessible to more threat agents. Different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

Note: For the following threats, a simple enrolment process by sending the template from the same sensor as verification to template storage is being assumed.

Threat 17 Tamper with feature extractor process (software) to capture a legitimate template and replay it later on. This attack focuses on IO (Input/Output) functions of the process. Depending on software design, IO functions may or may not be easier to tamper with. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much more flexible time constraints.

Threat 18 Capture template transmission of a legitimate user from feature extractor to template storage and replay it later on. This also requires the relevant identification process result. This is the case for both static and mobile readers. The mobile nature of mobile devices provides threat agents with much easier physical constraints, since the data flow has to use some type of wireless technology, possibly including intermediate communication devices. In such a case, the data flow itself is a complex element (not to be mistaken with complex process) and can be broken down for more granular DFD levels. Based on the location of the template storage, different threat agents have advantage to perform this attack. Refer to Table 6.1 for relevant threat agents in each scenario.

7 Results of threat analysis

The results included here are overall insights, based on combined evaluation of data at hand. Insights specific to scenarios with smaller domain of impact are considered for Subsection 7.6. Remember that this is a high-level analysis and these insights have to considered in the context of actual use-cases and implementations.

7.1 Methodology

We have explained in Subsection 3.7 that the threat modelling process includes iterative improvements. In our experience, a lower level (more detailed) threat model can be used as a feed to return and improve the previous higher level (less detailed) one. After creating the attack tree representation in our analysis, we have switched to STRIDE threat listing. This transition proved to be extremely efficient in improving the attack tree by revealing misconceptions and missed attack approaches. For instance, the addition of attack vector 5 as a leaf under *spoof temporary artefacts* and *spoof persistent artefacts* in Figure 6.2 was based on the results from Table 6.1, namely, TH10.1, TH10.2, TH11.1 and TH11.2. This is a direct result of introducing details in the form of metadata and extending the standard STRIDE listing into Table 6.1, which is our recommended approach. The resulting overview is superior to just attack trees, or just STRIDE listings, as it provides easily derivable quantitative information, along with the qualitative information.

As for the threats listed, we have considered commonly known potential threat against the leafs of the attack tree. One can always come up with more threats, but we must take into consideration that this is just one iteration in the whole process and the goal is the gain enough insights to be able to move to the next, more detailed iteration.

7.2 Statistics

Without considering the conductibility of these threats and simply by focusing on the statistical big picture, interesting results are observable. From a pool of 30 threats presented in Table 6.1, scenarios 1 to 4 are susceptible to 18 threats, each. Table 7.1 provides the number of threats by different threat agents in each scenario, as well as the sum of threats by them.

These numbers can be considered as one of the metrics for acceptance, or rejection of scenarios and consequently, their architecture. For instance, knowing that there can be less control over what a carrier, or an attacker does, we can assume scenario 3 as having the highest

Threat	Vector	Element	Agent(s)	Artefact	S	T	R	I	D	E	Scenario(s)
TH1	1	mobile device	carrier, attacker	attribute	×			∗			S1, S2, S3, S4
TH2	1	mobile device	carrier, attacker	attribute	×	∗					S1, S2, S3, S4
TH3	2	mobile device	attacker	raw data	×			∗			S1, S2, S3, S4
TH4	2	mobile device	carrier, attacker	raw data	∗						S1, S2, S3, S4
TH5	3	mobile device	carrier, attacker	sample	×	∗					S1, S2, S3, S4
TH6	3	mobile device	attacker	sample	×	∗		×			S1, S2, S3, S4
TH7.1	4	sample data flow	attacker, administrator	sample	×			∗			S1, S2, S4
TH7.2	4	sample data flow	attacker	sample	×			∗			S3
TH8.1	4	sample data flow	carrier, attacker, administrator	sample	∗			×			S1, S2, S4
TH8.2	4	sample data flow	carrier, attacker	sample	∗			×			S3
TH9.1	5	biometric verifier	administrator	result		∗					S1, S2, S4
TH9.2	5	mobile device	carrier, attacker	result		∗					S3
TH10.1	5	biometric verifier	administrator	sample	×	∗		×			S1, S2, S4
TH10.2	5	mobile device	carrier, attacker	sample	×	∗		×			S3
TH11.1	5	biometric verifier	administrator	template	×	∗		×			S1, S2, S4
TH11.2	5	biometric verifier	carrier, attacker	template	×	∗		×			S3
TH12.1	6	template storage	carrier, attacker	template	×			∗			S1, S3
TH12.2	6	template storage	administrator	template	×			∗			S2, S4
TH13.1	7	query data flow	attacker	template	×			∗			S1, S3
TH13.2	7	query data flow	administrator	template	×			∗			S2, S4
TH14.1	7	template data flow	attacker, administrator	template	×			∗			S1
TH14.2	7	template data flow	administrator	template	×			∗			S2, S4
TH14.3	7	template data flow	attacker	template	×			∗			S3
TH15.1	8	result data flow	administrator	result	×			∗			S1, S2
TH15.2	8	result data flow	attacker, administrator	result	×			∗			S3. S4
TH16.1	8	result data flow	administrator	result	×	∗					S1, S2
TH16.2	8	result data flow	attacker, administrator	result	×	∗					S3, S4
TH17	9	mobile device	attacker	template	×	∗					S1, S2, S3, S4
TH18.1	9	template data flow	attacker	template	×			∗			S1, S3
TH18.2	9	template data flow	attacker, administrator	template	×			∗			S2, S4

Table 6.1: A threat list, including all relevant metadata, i.e. related attack vector under biometrics, involved element in DFDs, involved agents, artefact at risk, primary (∗) and secondary (×) STRIDE categories and related scenarios.

Threat agent	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Sum
Carrier	6	5	9	5	25
Attacker	13	10	18	12	53
Administrator	8	11	2	11	32

Table 7.1: Number of threats by each threat agent in different scenarios

risk. This is also backed by the sum of threats by each threat agent, where we can see more threats by carriers and especially attackers are possible in scenario 3. In a sense, the effectiveness of a threat agent is also being monitored.

Another insight that can be backed up by these numbers is the high risk of transferring artefacts over external boundaries. Having an artefact on both internal and external realms would mean both internal threat agents (administrators) and external threat agents (carriers, attackers) will have access to it. This is clearly reflected in scenarios 1, 2 and 4, where we have a relatively even spread of threats in relation with threat agents (attackers and administrators), suggesting equal opportunities by internal and external threat agents. As a result, it is advisable to focus more on administrator privileges and threats over communication links, in scenarios with high amount of unnecessary transmission of sensitive artefacts over external boundaries.

7.3 Elements and artefacts

It is clear from Table 6.1 that the majority of threats involve *information disclosure*, or *tampering*, as their primary STRIDE category. Also, most of these have data flows and data stores as their target elements. Template, as the artefact under focus, has the highest occurrence as well. Considering that a template is a persistent artefact, kept in the system for long-term, these observations reiterate the need for template protection mechanisms.

7.4 Communication channels

The prevalence of information disclosure in data flows, along with the fact that replayed transmissions could result in breaches, promotes the importance of confidentiality and integrity. When it comes to data flows, this can be achieved in the form of secure virtual channels. Regardless of involved paths, or nodes in the communication channel, there has to be a direct and secure virtual end-to-end channel between the provider element and the receiving element. We must elaborate that a receiving element, especially when it comes to sensitive biometric artefacts, is an element with some form of processing functionality. This statement means, unless an artefact has to be processed by an element for authentication purposes, the artefact must be kept confidential over the end-to-end channel. Pay attention that this is regardless of the template protection mechanism in place.

7.5 Privacy implications

Apart from the aforementioned security related attack vectors, when it comes to persistent biometric artefacts, we should also pay attention to privacy considerations. This is especially important for identification using biometrics, since persistent biometric artefacts, such as a

template, are valid permanently. To guarantee the privacy of transactions carried out by a validated user, the underlying systems and the validation mechanism must be unaware towards the user's identity. This consideration alone, requires the use of privacy preserving template protection schemes, fulfilling the requirements mentioned in Subsection 3.4.

7.6 Mitigation

It is important to point out that when it comes to mitigation, we are not carrying out any form of risk management. We are especially interested in what mitigation potential is provided by orthodox mitigation mechanisms for introduced STRIDE threats, as well as biometric template protection schemes. We also would like to understand which schemes are more suitable for our BYOD categories, BYOAuthenticator and BYOAuthentication. We will also include more to the point and scenario specific examples.

7.6.1 Common mitigations

We can divide mitigation mechanisms into biometric template protection schemes, and the rest. When dealing with information disclosure, tampering and spoofing, mitigation methods rely on the correct usage of confidentiality and integrity protection. In other words, encryption and signing. We have mentioned the need for secure end-to-end channels. To be more specific, whenever an artefact, whether temporary or persistent, is transmitted between a generating element and a processing element, the channel has to be encrypted. This applies to non-biometric artefacts, such as result, too. For instance, a biometric sample must be transmitted over an encrypted channel to a biometric verifier. Same applies to the template transmission from template storage to biometric verifier and result transmission from biometric verifier to controller. Similarly, different data stores must be secured using encryption. The most important ones are template storage and log storage.

Remember that just encrypting the network traffic does not address tampering or spoofing. To make sure that the traffic is from the genuine source, a Public-Key Infrastructure (PKI) system has to be in place, organisation-wide. This means that, especially the mobile device has to be known using a certificate, suggesting the need for a device enrolment process. To reiterate, even in BYOAuthentication scenarios (S3 and S4), although authentication process is external, the mobile device has to be known and enrolled with the organisation, since the result is transmitted from it.

Attacks such as hill-climbing, which rely on the information retrieved from the trial and error process as a feed to their modification scheme, can be limited through *intrinsic security*, as mentioned in Subsection 3.4, and logging mechanisms. The result of an unsuccessful verification should not be transmitted unless there is a real need for it. This verification score should not be sent back to carrier and a message informing the user of negative result must not include the score. The other important factor for these attacks is log storage access. Especially with third-parties, there has to be means of access to event logs, otherwise, scenarios including third-parties are not recommended. Considering this and other mitigations, one can define an acceptable criteria for third-party biometric verifiers. This applies to both BYOauthentication, where the carrier uses an external service, or BYOAuthenticator, if the organisation uses an external biometric verifier. A trust scale using different Trusted Secure (TS) and Trusted Private (TP)

levels, as introduced in Section 4, can be defined based on the mitigation functionality provided by the third-party biometric verifier.

To prevent process-level tampering, all critical functionality such as matching, must be executed in Secure Access Modules, or similar tamper-proof devices. This requirement applies to all cases. As a result, scenario 3, with biometric verification on the mobile device, is not a suitable one. Even if the mobile device is employing a secure zone for processes and storage, it is only accessible to the manufacturer. Unless there is a tamper-proof and accessible area available to implementers, scenario 3 must be avoided.

Considering all the risks involved with biometric artefacts, especially privacy related ones, a template protection scheme must be in place on top of everything else. Given the three possibilities of central storage (PI and AD stored in the organisational database), mobile storage (PI and AD stored on mobile device) and hybrid storage (AD on mobile device, PI in the organisational database, or vice versa), schemes can be utilised. The important special case to describe here is when the requirements dictate the storage of template in a token with the carrier. In such a case, instead of having a mobile storage (mobile device as a token), hybrid storage has to be used. AD part can be stored on the mobile device, while PI is on another form of protected storage, such as a smart card. If the verification process is also handled by the mobile device, a PIV approach or a somewhat homomorphic encryption-based one, as discussed in Subsection 3.4, can be beneficial. Although, this assumes the existence of a tamper-proof secure area for processing as explained before. The actual choice of the template protection scheme is out of the scope of this study.

7.6.2 Specific mitigations

Here, we will go through some of the important mitigation requirements for different scenarios and motivations behind them. Table 7.2 presents these.

Scenario 1 A PIR based template protection scheme may be utilised. This is due to the separation of template storage and biometric verifier. A secure element on the mobile device is also required to store the template, which is AD in a hybrid storage case.

Scenario 2 Either a PIV, or a SWHE based template protection scheme may be utilised. This is due to the unified structure of biometric verifier and template storage. Theoretically speaking, a SWHE, encrypting the transmission of sample and result, is preferable.

Scenario 3 A PIV based template protection scheme may be utilised. This is due to the unified structure of biometric verifier and template storage. The result has to be sent in plain-text to the central server. Thus a SWHE based solution is not required, assuming there is a secure element available for the PIV process. The result can be sent with standard encryption and signature based on PKI. A secure element on the mobile device is also required to store the template and run the verification process.

Scenario 4 A SWHE based template protection scheme may be utilised. This is due to the unified structure of biometric verifier and template storage, as well as it being on a third-party. A trust scale for third-party evaluation may be required.

Mitigation	Scenario 1	Scenario 2	Scenario 3	Scenario 4
PIR	×			
PIV		×	×	
SWHE		×		×
Secure element (process)			×	
Secure element (storage)	×		×	
Trust scale				×

Table 7.2: Important mitigations per scenario

8 Conclusion and future work

Going back to the research question, in this report, we have provided a methodology for high-level threat analysis of a PACS, involving biometrics. We have gone through the architectural differences introduced with integration of mobile devices in such an environment, based on the scenarios and requirements dictated by that integration. Considering the threat analysis methodology, what it boils down to is that building an extended STRIDE threat list based on the combination of DFDs, attack trees and scenarios is highly beneficial as it provides the details needed for the next improving iteration of the analysis. We called this process an architectural threat analysis. Each step, resulting in the final extended STRIDE threat list has its part to play. Scenarios help us to limit our problem space and to have contained input to work with, otherwise, one has to consider every architecture and every set-up. DFDs make it possible to create simplified abstractions for architectures describing scenarios. The resulting extended STRIDE threat list is an effective visualisation of the data at hand, showing and linking many details at once. Finally, attack trees act as a tool to divide and conquer the threat analysis problem. The same methodology can be applied to other complex systems, providing an efficient approach towards a secure-by-design solution.

During the process, we have introduced BYOAuthenticator and BYOAuthentication, as BYOD possibilities for a PACS-biometrics combination. These definitions, along with their included scenarios can be considered for other authentication and access control systems as well and are not specific to biometrics, or PACS per se.

Following other focus areas of the research question, we have also listed many high-level attacks focusing on the biometrics point of view and have included mitigation best-practices, reiterating the importance of template protection schemes. Our analysis has also generated simple statistics, based on the results presented in Table 6.1. Since quantitative metrics are quite valuable, it would be interesting to see more studies on the statistical result of such threat analyses. The next iteration of our threat analysis would be one level further in detail, generating more complex DFDs and more detailed attack trees. This will result in a larger table with more statistical raw data.

Secure, accessible storage and processing capabilities on mobile devices is another important missing link, which might not be trivial because of the dependence to hardware architecture. Since this is a limiting factor for our scenarios, a study of possibilities provided by NFC secure element on mobile devices and its comparison against Host Card Emulation (HCE), specifically for biometrics applications, will be highly beneficial. HCE allows mobile devices and their application to utilise NFC functionality without requiring a secure element by providing the

possibility of emulating smart cards and placing card data on a cloud.

References

[1] Apple. *iOS Security. iOS 9.0 or later*. Apple, 2015. URL: https://www.apple.com/business/docs/iOS_Security_Guide.pdf.

[2] ARM. *ARM Security Technology. Building a Secure System using TrustZone®Technology*. ARM Limited, 2009. URL: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.

[3] Marc Bouissou and Jean-Louis Bon. 'A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes'. In: *Reliability Engineering & System Safety* 82.2 (2003), pp. 149–163. ISSN: 0951-8320. DOI: http://dx.doi.org/10.1016/S0951-8320(03)00143-1. URL: http://www.sciencedirect.com/science/article/pii/S0951832003001431.

[4] Jeroen Breebaart et al. 'A reference architecture for biometric template protection based on pseudo identities'. In: *In BIOSIG 2008: Biometrics and Electronic Signatures*. 2008.

[5] Julien Bringer et al. 'Extended Private Information Retrieval and Its Application in Biometrics Authentications'. In: *Proceedings of the 6th International Conference on Cryptology and Network Security*. CANS'07. Singapore: Springer-Verlag, 2007, pp. 175–193. ISBN: 3-540-76968-4, 978-3-540-76968-2. URL: http://dl.acm.org/citation.cfm?id=1778554.1778569.

[6] Common Criteria. *Common Criteria for Information Technology Security Evaluation (CC v3.1 r4)*. National Information Assurance Partnership (NIAP), 2012. URL: https://www.niap-ccevs.org/Documents_and_Guidance/cc_docs/CCPART1V3.1R4.pdf.

[7] Yevgeniy Dodis, Leonid Reyzin and Adam Smith. 'Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data'. In: *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings*. Ed. by Christian Cachin and Jan L. Camenisch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 523–540. ISBN: 978-3-540-24676-3. DOI: 10.1007/978-3-540-24676-3_31. URL: http://dx.doi.org/10.1007/978-3-540-24676-3_31.

[8] Craig Gentry. 'A Fully Homomorphic Encryption Scheme'. AAI3382729. PhD thesis. Stanford, CA, USA, 2009. ISBN: 978-1-109-44450-6.

[9] Ida Hogganvik and Ketil Stølen. 'A Graphical Approach to Risk Identification, Motivated by Empirical Investigations'. In: *Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems*. MoDELS'06. Genova, Italy: Springer-Verlag, 2006, pp. 574–588. ISBN: 3-540-45772-0, 978-3-540-45772-5. DOI: 10.1007/11880240_40. URL: http://dx.doi.org/10.1007/11880240_40.

[10] B. Hulsebosch, G. Lenzini and H. Eertink. *D2.3 - Quality authenticator scheme*. STORK, 2009-03-03. URL: https://www.eid-stork.eu/dmdocuments/public/D2.3_final._1.pdf.

[11] *ISO/IEC 24745:2011. Information technology – Security techniques – Biometric information protection*. International Organisation for Standardisation (ISO), 2011-06-15. URL: http://www.iso.org/iso/catalogue_detail?csnumber=52946.

[12] Anil K. Jain, Ruud Bolle and Sharath Pankanti. *Biometrics: Personal Identification in Networked Society*. Springer Publishing Company, Incorporated, 1999. ISBN: 1475782942, 9781475782943.

[13] Anil K. Jain, Karthik Nandakumar and Abhishek Nagar. 'Biometric Template Security'. In: *EURASIP J. Adv. Signal Process* 2008 (2008-01), 113:1–113:17. ISSN: 1110-8657. DOI: 10.1155/2008/579416. URL: http://dx.doi.org/10.1155/2008/579416.

[14] Anil K. Jain and Arun Ross. 'Introduction to Biometrics'. In: *Handbook of Biometrics*. Ed. by Anil K. Jain, Patrick Flynn and Arun A. Ross. Boston, MA: Springer US, 2008, pp. 1–22. ISBN: 978-0-387-71041-9. DOI: 10.1007/978-0-387-71041-9_1. URL: http://dx.doi.org/10.1007/978-0-387-71041-9_1.

[15] A. Juels and M. Sudan. 'A fuzzy vault scheme'. In: *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*. 2002, pp. 408–. DOI: 10.1109/ISIT.2002.1023680.

[16] Ari Juels and Martin Wattenberg. 'A Fuzzy Commitment Scheme'. In: *Proceedings of the 6th ACM Conference on Computer and Communications Security*. CCS '99. Kent Ridge Digital Labs, Singapore: ACM, 1999, pp. 28–36. ISBN: 1-58113-148-8. DOI: 10.1145/319709.319714. URL: http://doi.acm.org/10.1145/319709.319714.

[17] Barbara Kordy et al. 'Foundations of Attack-defense Trees'. In: *Proceedings of the 7th International Conference on Formal Aspects of Security and Trust*. FAST'10. Pisa, Italy: Springer-Verlag, 2011, pp. 80–95. ISBN: 978-3-642-19750-5. URL: http://dl.acm.org/citation.cfm?id=1964555.1964561.

[18] Jean-Paul Linnartz and Pim Tuyls. 'New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates'. In: *Proceedings of the 4th International Conference on Audio- and Video-based Biometric Person Authentication*. AVBPA'03. Guildford, UK: Springer-Verlag, 2003, pp. 393–402. ISBN: 3-540-40302-7. URL: http://dl.acm.org/citation.cfm?id=1762222.1762276.

[19] *Microsoft Security Development Lifecycle*. Microsoft. 2016. URL: https://www.microsoft.com/en-us/SDL (visited on 2016-07-25).

[20] Fabian Monrose, Michael K. Reiter and Susanne Wetzel. 'Password Hardening Based on Keystroke Dynamics'. In: *Proceedings of the 6th ACM Conference on Computer and Communications Security*. CCS '99. Kent Ridge Digital Labs, Singapore: ACM, 1999, pp. 73–82. ISBN: 1-58113-148-8. DOI: 10.1145/319709.319720. URL: http://doi.acm.org/10.1145/319709.319720.

[21] C. Moore et al. 'Practical homomorphic encryption: A survey'. In: *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2014-06, pp. 2792–2795. DOI: 10.1109/ISCAS.2014.6865753.

[22] Suvda Myagmar, Adam J. Lee and William Yurcik. 'Threat Modeling as a Basis for Security Requirements'. In: *In Symposium on Requirements Engineering for Information Security (SREIS)*. 2005.

[23] Committee on National Security Systems. *National Information Assurance (IA) Glossary*. Committee on National Security Systems, 2010. URL: https://www.ncsc.gov/nittf/docs/CNSSI-4009_National_Information_Assurance.pdf.

[24] *ONVIF Profile C Specification v1.0*. ONVIF, 2013. URL: http://www.onvif.org/Portals/0/documents/specs/2013_12_ONVIF_Profile_C_Specification_v1-0.pdf.

[25] Mano Paul. *Official (ISC)2 Guide to the CSSLP*. 1st. Boca Raton, FL, USA: CRC Press, Inc., 2011. ISBN: 1439826056, 9781439826058.

[26] N. Poolsappasit, R. Dewri and I. Ray. 'Dynamic Security Risk Management Using Bayesian Attack Graphs'. In: *IEEE Transactions on Dependable and Secure Computing* 9.1 (2012-01), pp. 61–74. ISSN: 1545-5971. DOI: 10.1109/TDSC.2011.34.

[27] Salil Prabhakar, Sharath Pankanti and Anil K. Jain. 'Biometric Recognition: Security and Privacy Concerns'. In: *IEEE Security and Privacy* 1.2 (2003-03), pp. 33–42. ISSN: 1540-7993. DOI: 10.1109/MSECP.2003.1193209. URL: http://dx.doi.org/10.1109/MSECP.2003.1193209.

[28] N. K. Ratha, J. H. Connell and R. M. Bolle. 'Enhancing Security and Privacy in Biometrics-based Authentication Systems'. In: *IBM Syst. J.* 40.3 (2001-03), pp. 614–634. ISSN: 0018-8670. DOI: 10.1147/sj.403.0614. URL: http://dx.doi.org/10.1147/sj.403.0614.

[29] Nalini K. Ratha et al. 'Generating Cancelable Fingerprint Templates'. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29.4 (2007-04), pp. 561–572. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1004. URL: http://dx.doi.org/10.1109/TPAMI.2007.1004.

[30] Christian Rathgeb and Andreas Uhl. 'A survey on biometric cryptosystems and cancelable biometrics'. In: *EURASIP Journal on Information Security* 2011.1 (2011), pp. 1–25. ISSN: 1687-417X. DOI: 10.1186/1687-417X-2011-3. URL: http://dx.doi.org/10.1186/1687-417X-2011-3.

[31] Keunwoo Rhee et al. 'Threat Modeling of a Mobile Device Management System for Secure Smart Work'. In: 13.3 (2013-09), pp. 243–256. ISSN: 1389-5753. DOI: 10.1007/s10660-013-9121-4. URL: http://dx.doi.org/10.1007/s10660-013-9121-4.

[32] Ronald L. Rivest, Len Adleman and Michael L. Dertouzos. 'On data banks and privacy homomorphisms'. In: *Foundations of secure computation* 4 (11 1978), pp. 169–180.

[33] Chris Roberts. 'Biometric Attack Vectors and Defences'. In: *Comput. Secur.* 26.1 (2007-02), pp. 14–25. ISSN: 0167-4048. DOI: 10.1016/j.cose.2006.12.008. URL: http://dx.doi.org/10.1016/j.cose.2006.12.008.

[34] Paul Ruggiero and Jon Foote. *Cyber Threats to Mobile Phones*. US-CERT, 2011. URL: https://www.us-cert.gov/sites/default/files/publications/cyber_threats_to_mobile_phones.pdf.

[35] Bruce Schneier. *Attack Trees*. Schneier on Security, 1999. URL: https://www.schneier.com/cryptography/archives/1999/12/attack_trees.html (visited on 2016-04-12).

[36] Robert C. Seacord. 'Mobile Device Security'. In: *Proceedings of the 3rd International Workshop on Mobile Development Lifecycle*. MobileDeLi 2015. Pittsburgh, PA, USA: ACM, 2015, pp. 1–2. ISBN: 978-1-4503-3906-3. DOI: 10.1145/2846661.2846671. URL: http://doi.acm.org/10.1145/2846661.2846671.

[37] *Security Industry Association*. SIA. URL: http://www.securityindustry.org (visited on 2016-04-15).

[38] C. Senk and F. Dotzler. 'Biometric authentication as a service for enterprise identity management deployment: a data protection perspective'. In: *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*. 2011-08, pp. 43–50. DOI: 10.1109/ARES.2011.14.

[39] Adam Shane et al. *Physical Access Control System (PACS) in a Federal Identity, Credentialing and Access Management (FICAM) Framework. PACS Best Practices using PKI-Authentication*. Security Industry Association, 2013. URL: http://www.siaonline.org/SiteAssets/Standards/EPTWG_White%20Paper_v5.pdf.

[40] R. Shirey. *RFC 4949: Internet Security Glossary, Version 2*. IETF, 2007.

[41] Adam Shostack. *Threat Modeling: Designing for Security*. 1st. Wiley Publishing, 2014. ISBN: 1118809998, 9781118809990.

[42] Yulian Slobodyan. *Security Training: #3 Threat Modelling - Practices and Tools*. SoftServe, 2014. URL: http://www.slideshare.net/YulianSlobodyan/3threat-modelling-practices-and-tools (visited on 2016-04-19).

[43] Colin Soutar. 'Biometric System Security'. In: (2002).

[44] Colin Soutar et al. 'Biometric Encryption using image processing'. In: vol. 3314. 1998, pp. 178–188. DOI: 10.1117/12.304705. URL: http://dx.doi.org/10.1117/12.304705.

[45] Richard Stallman. *Who does that server really serve?* GNU Operating System, 2016. URL: https://www.gnu.org/philosophy/who-does-that-server-really-serve.html (visited on 2016-04-08).

[46] A. Stango, N. R. Prasad and D. M. Kyriazanos. 'A Threat Analysis Methodology for Security Evaluation and Enhancement Planning'. In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. 2009-06, pp. 262–267. DOI: 10.1109/SECURWARE.2009.47.

[47] *STORK*. STORK. 2012. URL: https://www.eid-stork.eu (visited on 2016-04-07).

[48] *Threat Risk Modeling*. OWASP, 2015. URL: https://www.owasp.org/index.php/Threat_Risk_Modeling (visited on 2016-04-12).

[49] Mark Turner, David Budgen and Pearl Brereton. 'Turning Software into a Service'. In: *Computer* 36.10 (2003-10), pp. 38–44. ISSN: 0018-9162. DOI: 10.1109/MC.2003.1236470. URL: http://dx.doi.org/10.1109/MC.2003.1236470.

[50] Pim Tuyls et al. 'Practical Biometric Authentication with Template Protection'. In: *Proceedings of the 5th International Conference on Audio- and Video-Based Biometric Person Authentication*. AVBPA'05. Hilton Rye Town, NY: Springer-Verlag, 2005, pp. 436–446. ISBN: 3-540-27887-7, 978-3-540-27887-0. DOI: 10.1007/11527923_45. URL: http://dx.doi.org/10.1007/11527923_45.

[51] U. Uludag and A. K. Jain. 'Attacks on biometric systems: a case study in fingerprints'. In: *Security, Steganography, and Watermarking of Multimedia Contents VI*. Ed. by E. J. Delp III and P. W. Wong. Vol. 5306. Proceedings of the International Society for Optical Engineering. 2004-06, pp. 622–633. DOI: 10.1117/12.530907.

[52] Timothy Vidas, Daniel Votipka and Nicolas Christin. 'All Your Droid Are Belong to Us: A Survey of Current Android Attacks'. In: *Proceedings of the 5th USENIX Conference on Offensive Technologies*. WOOT'11. San Francisco, CA: USENIX Association, 2011, pp. 10–10. URL: http://dl.acm.org/citation.cfm?id=2028052.2028062.

[53] Fredrik Vraalsen et al. 'Specifying Legal Risk Scenarios Using the CORAS Threat Modelling Language'. In: *Proceedings of the Third International Conference on Trust Management*. iTrust'05. Paris, France: Springer-Verlag, 2005, pp. 45–60. ISBN: 3-540-26042-0, 978-3-540-26042-4. DOI: 10.1007/11429760_4. URL: http://dx.doi.org/10.1007/11429760_4.

[54] James L. Wayman et al. *Biometric Systems: Technology, Design and Performance Evaluation*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN: 1852335963.

[55] Masaya Yasuda et al. 'Packed Homomorphic Encryption Based on Ideal Lattices and Its Application to Biometrics'. In: *Security Engineering and Intelligence Informatics: CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings*. Ed. by Alfredo Cuzzocrea et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 55–74. ISBN: 978-3-642-40588-4. DOI: 10.1007/978-3-642-40588-4_5. URL: http://dx.doi.org/10.1007/978-3-642-40588-4_5.

[56] Zamboni. *Attacking Biometric Access Control Systems*. DEF CON, 2005. URL: https://www.defcon.org/images/defcon-13/dc13-presentations/DC_13-Zamboni.pdf (visited on 2016-04-28).