ERNW RESEARCH
pursuing knowledge.

ERNW WHITE PAPER 70

HL7®FHIR®: PRESERVING DISTRIBUTED RESOURCE INTEGRITY

Version:	1.0
Date:	December 17, 2020
Classification:	Public
Author(s):	Julian Suleder, Nina Matysiak

Julian Suleder

Digital unterschrieben von Julian Suleder
Datum: 2020.12.17 20:31:16 +01'00'

ERNW RESEARCH
pursuing knowledge.

Table of Content

ERNW RESEARCH
pursuing knowledge.

List of Figures

ERNW RESEARCH
pursuing knowledge.

List of Abbreviations

API	Application Programming Interface
ASTM	American Society for Testing and Materials
CDA	Clinical Document Architecture
EHR	Electronic Health Record
FHIR®	Fast Healthcare Interoperability Resources
HL7®	Health Level Seven
HTML	Hypertext Markup Language
IETF	Internet Engineering Task Force
JSON	JavaScript Object Notation
JWK	JSON Web Key
JWS	JSON Web Signature
JWT	JSON Web Token
MIME	Multipurpose Internet Mail Extensions
REST	Representational State Transfer
SOA	Service Oriented Architecture
UDI	Unique Device Identifier
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

ERNW RESEARCH pursuing knowledge.

1 Introduction

This work assesses the HL7® FHIR® standard regarding its statements on integrity and accountability.

1.1 Motivation

Digital networking is already widespread in many areas of life. More and more medical devices are also being networked in the healthcare industry. This growth makes the development and use of new medical communication standards necessary since existing solutions can only meet the changing requirements with great effort. The Health Level Seven (HL7®) Fast Healthcare Interoperability Resources (FHIR®) standard is an example of such a medical communication standard. FHIR® is said to have increased the interoperability between different medical contexts,e.g., administration, billing, and clinical care, to enable data exchange of various systems. FHIR® tries to solve this with direct access to information fields as an alternative to document-centered approaches.

The FHIR® standard addresses the security risks associated with strongly networked communication from a large number of systems across the trust and organizational boundaries only indirectly because FHIR® does not define mandatory security controls or requirements. The standard refers to the use of state-of-the-art methods and security controls, including common security standards and web technologies,e.g., authentication, authorization, and access control.

In a clinical context in which FHIR® is used for the exchange of sensitive data between different systems and the transfer of data across organizational units and the boundaries of healthcare providers, it must be assumed that compromised systems are also involved in the communication. When medical decisions are made based on manipulated data, this poses a high risk of threatening patient safety.

1.2 Objective

In this paper, we analyze the security controls of the HL7® FHIR® communication standard that is concerned with non-repudiation and accountability of resource modifications and the integrity of exchanged resources. To do this, existing FHIR® resource types are examined with a focus on answering the following questions:

- How can the existing FHIR® *Provenance* and *Signature* resources be used to verify and ensure the identity of resources created and modified in a distributed system during and after the communication sequence?
- To what extent is it possible to identify all other systems that have requested and processed this data?
- Which requirements does the preservation of resource integrity pose on security measures to be implemented?
- Which parts of the FHIR® standard are required to ensure non-repudiation and integrity?

This paper aims to discuss the current functionality and improve the HL7® FHIR® communication standard. The objective is not to develop a proof of concept in the form of a demonstrator or to show instructions for building FHIR® systems.

1.3 Outline

Here, we will explicate the structure of this document. After the introduction is provided in the present section, Section 2 focuses on the basics of Fast Healthcare Interoperability Resources (FHIR®) and the technical fundamentals essential for this work. In Section 3, the concepts, technologies, and procedures used are presented by analyzing the standard and the existing resource types *Provenance* and *Signature*. Based on the findings of the analysis, the concept for assuring distributed resource integrity is presented in Section 4. Section 5 concludes this work with a discussion of the results and an outlook for further research.

2 Terms & Concepts

This section introduces the basic concepts needed to discuss a FHIR® system in the context of this paper. At first, in Section 2.1 basic security goals are presented. Afterward, a brief introduction to trust boundaries is given in Section 2.2. In Section 2.3 fundamentals and design principles of FHIR® are presented.

2.1 Security Goals

The following sections set out the fundamental security goals for this white paper.

2.1.1 Confidentiality

Confidentiality of a communication system is the protection against disclosure of information to unauthorized parties (Bundesamt für Sicherheit in der Informationstechnik (BSI), n.d.). To maintain confidentiality, security controls to establish and control information flows are required. These controls ensure that only authorized parties can access the data. The confidentiality of data can be ensured by, for example, using transport encryption. The data is transformed so that only the intended parties can meaningfully interpret the data (Bedner & Ackermann, 2010, pp. 323 sq.).

2.1.2 Integrity

The loss of integrity means that parts of the data or information have been replaced, expanded, or deleted (Bundesamt für Sicherheit in der Informationstechnik (BSI), n.d.; Bedner & Ackermann, 2010, p. 326) by unauthorized parties. To ensure integrity, protection measures to prevent such modifications must be implemented. Integrity can also be related to maintaining a temporal order of messages, as information may need to be received in a specific order to keep information content consistent (Bedner & Ackermann, 2010, pp. 326 sq.).

In the context of this paper, ensuring integrity means that communication channels must guarantee the integrity of the transferred data for communication relationships. This requirement indicates that secure variants such as HTTPS must be used at a communication protocol level in favor of their less secure alternatives. At the resource level of a FHIR® server, this means that integrity checks must be present and strict. A means of ensuring data integrity, even after transmission, is provided by FHIR® signatures with *Provenance* resources. Maintaining data integrity also includes a transactionality of the exchanged data to preserve the order of changes. The FHIR® servers should warrant this by appropriate concepts, such as optimistic locking. Some of these requirements may already be guaranteed by using architectural and implementational concepts such as the REST principles.

2.1.3 Availability

The availability of information technology systems, as well as the data and information processed therein, is given if the communication systems are ready for operation at any time, and data and information can be accessed as intended (Bundesamt für Sicherheit in der Informationstechnik (BSI), n.d.; Bedner & Ackermann, 2010, p. 326). One measure to increase availability is the use of redundant services.

In the context of this paper, the communication system should be resistant to failures of individual system components. On the one hand, this can be achieved through redundancy and tolerance towards failures. An example is that a medical device must preserve its medical functionality when a FHIR® server to which measured values are to be transmitted is not available. One example is the provision of redundant FHIR® servers so that even if one server fails, a transmission of, for example, measured values or querying patient information is possible. On the other hand, in the case of a network problem, however, this would often not be sufficient, so that the device should be able to cache measured values for later transmission.

2.1.4 Transparency

Transparency is partially the opposite of confidentiality since transparency means clarity, recognizability, and traceability. The transparency of systems demands that their structure is transparent and that their function and work are understandable. Besides, the data processed therein and the people involved and their actions should be recognizable. For example, it may be desirable or necessary for actions to be traceable or for communication to be observed and uncovered (Bedner & Ackermann, 2010, p. 325). This transparency does not mean that message content must be communicated transparently so that even transparent systems can be confidential.

In the context of this paper, this requirement implies that it is possible to name a list of all devices, user objects, services, and certificates present in the communication system to enable traceability of persons, services, and devices. This traceability is a compromise between anonymity and confidentiality of communication relationships but does not imply that the communication content must be transparent. Furthermore, the described information may not be generally accessible in the environment but partially known to distinct and distributed entities.

2.1.5 Authenticity

The term authenticity is used to describe the property that ensures, through appropriate control measures, that data and information come from the specified source and that the given identity of a user or a connected service is correct by providing a respective proof (Bundesamt für Sicherheit in der Informationstechnik (BSI), n.d.; Bedner & Ackermann, 2010, pp. 325 sq.).

In the context of this paper, the communication system should allow proving the identity of all communication partners. This can be achieved, for example, when end systems authenticate users against a central user administration. Where

possible, administrative users used for the configuration or maintenance of the devices can be authenticated against the central user administration, too. Besides being able to log authentication attempts centrally, the *Device* resource may need to be updated in the means of properties such as used software version or configured device capabilities. Validation of the identity of the changes' initiator may only be possible using the central authentication service.

2.1.6 Accountability and Non-Repudiation

In the case of accountability and non-repudiation, these security goals focus on verifiability to third parties by specific commitments. This verifiability, therefore, has a direct connection to transparency, authenticity, and integrity. A commitment aims to ensure that a sender cannot deny sending and that a recipient cannot deny receiving a specific message in a communication relationship. This commitment is also called non-repudiation of origin and non-repudiation of receipt. The message being indistinguishably attributable to the sender is also called accountability. For this to be possible, the properties of the sender, receiver, and message's authenticity and integrity must always be present. Accountability can also be necessary when separated from a communication relationship. For example, an IT system may require any change to a resource to be uniquely assigned to a subject. Besides, the subject would like to guarantee the authorship of exactly this change and the integrity of the changed data or information. The liability enables the subject to be attributed and the change to be demonstrated. Measures to guarantee the liability, especially the accountability, are the audit-proof logging of all actions and the use of digital signatures as well as time stamps (Bundesamt für Sicherheit in der Informationstechnik (BSI), n.d.; Bedner & Ackermann, 2010, pp. 325 sq.).

In the context of this paper, the system should guarantee the non-repudiation of sending a request to a FHIR® server and the non-repudiation of receipt. This can be done by using an *AuditEvent* resource. Within the framework of accountability, the identity of both communication partners must always be verified.

2.2 Trust Boundaries

When a system or system component interacts with other untrusted entities, there is a trust boundary to those entities. The term *Trust Boundary* is used primarily in threat modeling, such as the STRIDE model. The STRIDE model was developed by Microsoft for identifying and assessing threats (Shostack, 2014).

During threat modeling, a model of the system is created and analyzed. This model must be detailed enough to represent the components of the system and their communication relationships. In this model, trust boundaries and areas are defined based on the system model and assumptions where trust between systems is present and where not. A trust boundary describes an interface that allows the flow of data between different trust areas. Trust areas can be derived from the subset of data flows where the definition of the data and the actor do not correspond to the same component (Miller, 2008). A data flow that encompasses different trust areas must implicitly pass a trust boundary. All data that

crosses trust boundaries must be verified and secured. Otherwise, it is a Trust Boundary Violation as described in CWE-501 (MITRE, n.d.).

An example of a system model, including trust boundaries, is shown in Figure 1. The green area in the figure represents the area of high trust. The area outside is only partially trusted. The central service stores data in a database within its trust area and obtains information from environment variables and a configuration file outside its trust area. Furthermore, the service communicates with users or the network. These actors are also outside of their trust area. Any communication relationship with one of the components or actors outside the trusted area is via a trust boundary. The transition of data over such a boundary is marked with a gray circle. At this point, a strict validation of the entries must take place. On the other hand, there is no need for strict validation of the data from the service's database, as this source is trusted.

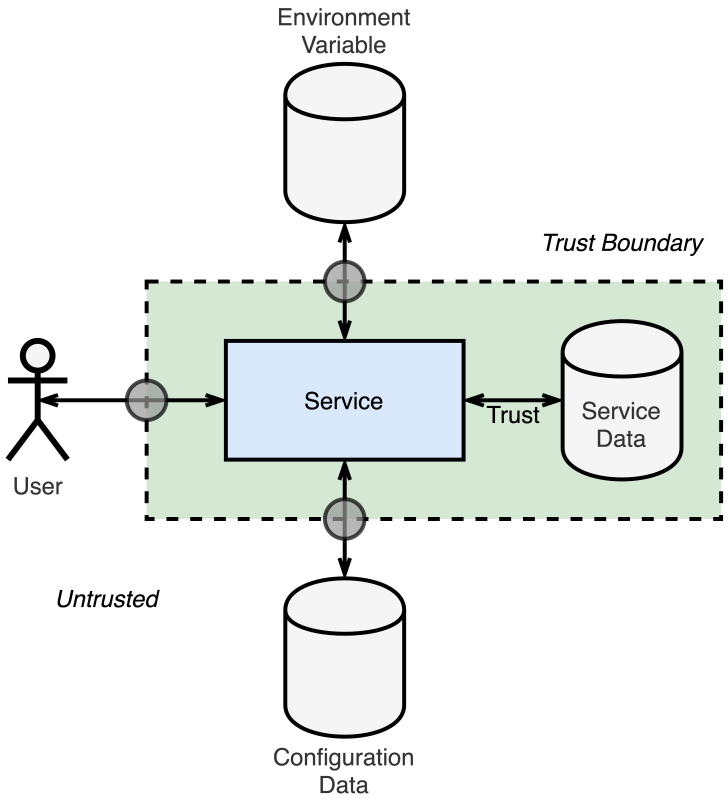

Figure 1: System Model Including Trust Boundaries

2.3 HL7® FHIR® Principles

Fast Healthcare Interoperability Resources (FHIR®) is a standard created by the Health Level Seven (HL7®) organization. The current version of FHIR® as of November 10, 2020, is R4 (Release 4 [v4.0.1: R4 - Mixed Normative and STU]), parts of the standard have reached the level *Normative*, while others are still at the level *Trial Use*. FHIR® builds on the experience of HL7® with the standards HL7® v2, HL7® v3 and Clinical Document Architecture (CDA). FHIR is based on modular components, so-called resources, and is based on the following principles:

- **Focus on implementers:**
  The target audience of the FHIR® standard consists of developers and other professional groups working directly on the implementation. In contrast to previous standards, such as HL7® v3, which followed a very structural and theoretical approach, a practical approach with focus on easy implementation is taken.
- **Focus on common scenarios:**
  One goal is to ensure that the FHIR® standard can be used in all healthcare areas worldwide. Given this breadth of application, wanting to map each use case natively would lead to an overload of the standard. That is why FHIR® focuses on supporting general scenarios following the so-called 80:20 rule. According to this rule, the aim is to define the building blocks which are called *resources* to cover 80% of the use cases natively. Simultaneously, the FHIR® standard offers the possibility of extensions for more specific use cases. Thus, data elements should only become part of the core specification if it is likely that most implementations will use this data element. The resources of the core specification should be as unchangeable and straightforward as possible.
- **Focus on established web technologies:**
  If possible, common web technologies such as XML, JSON, HTTP, and OAuth should be used to keep to keep the entry barriers for developers low and keep the FHIR® specification clean.
- **Focus on human-readable information:**
  All resources should have a human-readable part, the so-called narrative. This is to ensure that if technical interoperability fails, at least the display as Hypertext Markup Language (HTML) is possible.
- **Focus on openness and free availability:**
  The content of the FHIR® standard should be freely accessible and reusable, which is why it is published under the Creative Commons (CC) Public Domain license. This openness should contribute to a widespread use of FHIR®.
- **Support for different architectures and paradigms:**
  While the RESTful Application Programming Interface (API) of the FHIR® standard is the most widely used interface for the exchange of resources, the content of resources can also be exchanged using other architectures. For this FHIR® offers documents based on Clinical Document Architecture (CDA) or messages based on HL7® v2 and v3 and the support of Service Oriented Architectures (SOAs). Regardless of the architecture chosen, the exchanged content remains the same since FHIR® resources are always used. This paradigm also enables the exchange of FHIR® resources across architectural boundaries.

3 The FHIR® Environment

This section outlines the scenario in which the current functionality for preserving the resource integrity offered by FHIR® are applied in the means of the *Provenance* and *Signature* resources.

3.1 Identifying Resource Instances

A resource is an entity having an unique identity by which it can be referenced. The identity comprises of a set of structured data elements, hereafter called *attributes*. Resources have an identifiable version that must change as the content of the resource changes.

Each resource has an *id* element that contains the *logical ID* of the resource assigned by the server responsible for storage. It is unique within the scope of all the same resources on the same server and never changes after being assigned by the server. It is used to identify a resource in a system and to allow references between these resources. In doing so, resources refer either relatively or using an absolute URL composed of the server base address, the resource type, and the logical ID. FHIR® supports two kinds of references that support version information. These are called *literal references* and *canonical URLs*. Canonical URLs should be used to reference types of resources, whereas literal references should be used to reference specific instances of resources.

Most resources have a *master server*, the system on which the record was created first, and is managed. On the master server, the *id* may be chosen without much effort because the master server can assign and guarantee unique identifiers for its resources. This identifier choice is usually not a safe method for secondary systems that cannot enforce this identity consistency. As a result, when a resource is transferred, the copied entity does not always keep the same logical ID on the new server. This depends heavily on the implementation of the server or system. Therefore, if the logical ID were to change, all resource copies should reference the same underlying entity. This entity may also be in other formats,e.g., Clinical Document Architecture (CDA). Each representation, therefore, has uniform identifiers that consistently identify the entity. This is called *business identifier* which is stored in the *identifier* attribute and has the resource type *Identifier*. This resource defines the type of identifier that can be uniquely coded using public coding systems (Health Level Seven International (HL7), 2018b). The identifier can also be interpreted unambiguously between systems. In addition to the type, the identifier's specific coding system is defined in the form of a URI and the specific value of the identifier. Besides, information on a validity range of the identifier, an exhibitor, and a purpose of use can be given (Health Level Seven International (HL7), 2018g).

3.2 Referencing Resources

References are always defined as unidirectional. The corresponding inverse relationship from the target to the source is logical but is not usually explicitly represented in the target resource. For external references, navigating these reverse

relationships requires an external infrastructure to track the relationship between resources. A REST API provides such an infrastructure by providing the ability to find the inverse relationship by specifying search parameters for the references. Because resources are processed independently, relationships are not considered transitive.

The attributes *reference* and *identifier* of the *Reference* resource are used to denote literal identifiers of an *id* or an *identifier* (Health Level Seven International (HL7), 2018f). The type of the referenced resource can also be specified as a canonical URL using the *type* attribute. The URL points to the definition of the resource, as it may have been extended or sliced to meet application-specific or country-specific requirements.

3.3 Outline: Medical Device Scenario

In the following, a scenario of a doctor measuring vital signs with a patient monitor is presented. This scenario is modeled using FHIR®. The object diagram in Figure 2 shows a view on the relationship of the various realized instances of the resources. The necessary FHIR® resources are shown in the class diagram in Figure 3. The selection of resources and attributes is a subset of the attributes and resources defined in the FHIR® standard and is not exhaustive.

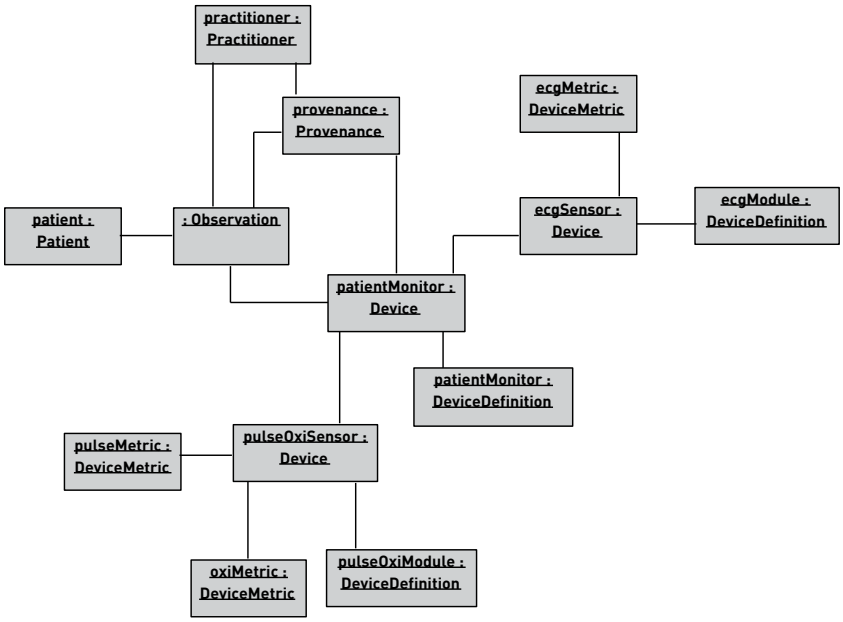

Figure 2: The object diagram shows the relationships of the instances used in the scenario.

In the class diagram in Figure 3, references are represented as Unified Modeling Language (UML) *Quantifier* instances. These references simplify the presentation resource references, and the owner of the reference responsible for maintaining referential consistency.

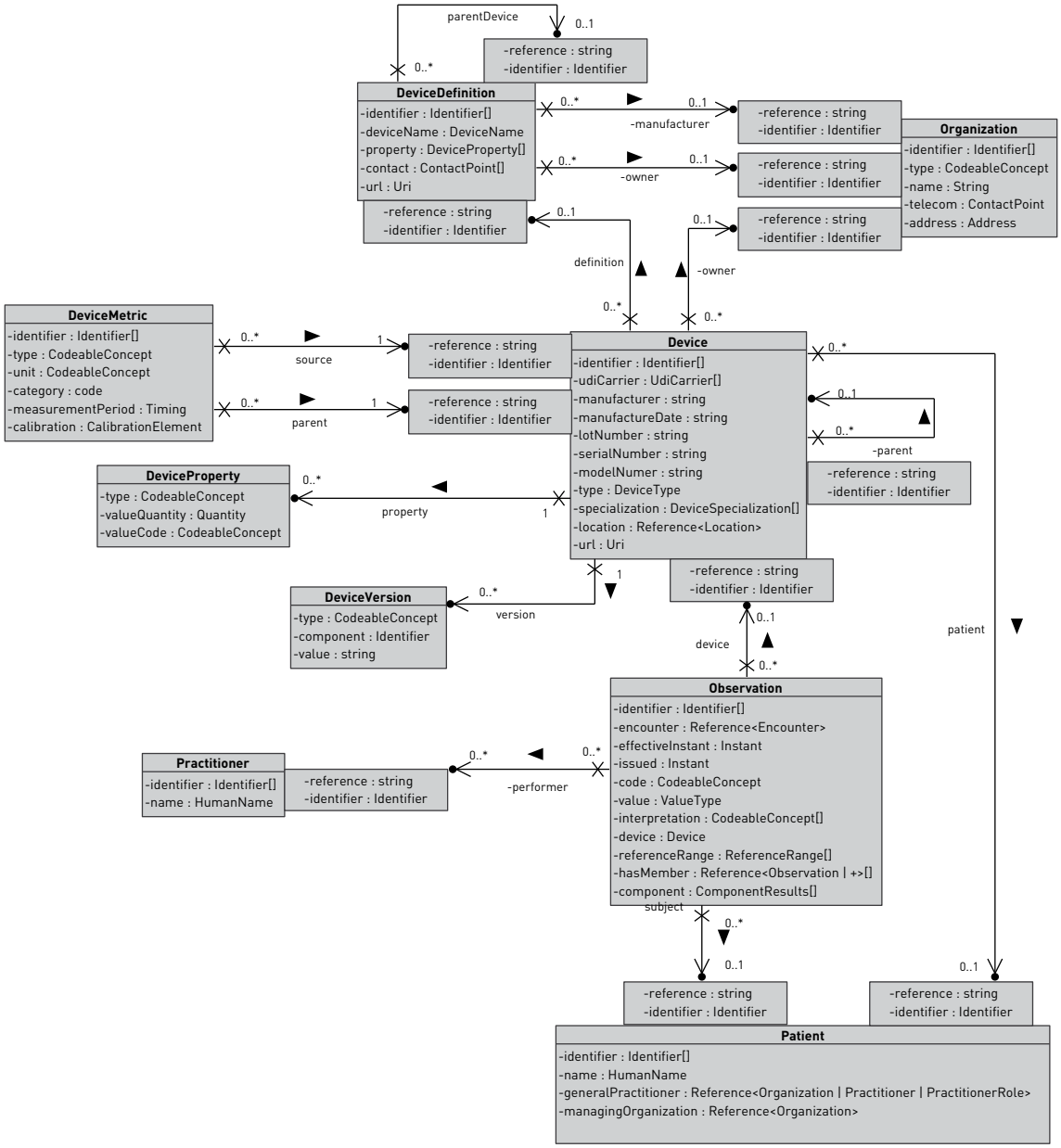

Figure 3: UML class diagram of the FHIR® resources relevant for the scenario, including references, cardinalities and attributes.

However, it cannot be assumed that these instances are available on the FHIR® server since the references can also refer to instance on remote systems. Though, the scenario assumes that all resources, with exception to instances of the resources *Observation* and *Provenance*, are persistent and available on a single FHIR® server. This is shown in Figure 4.

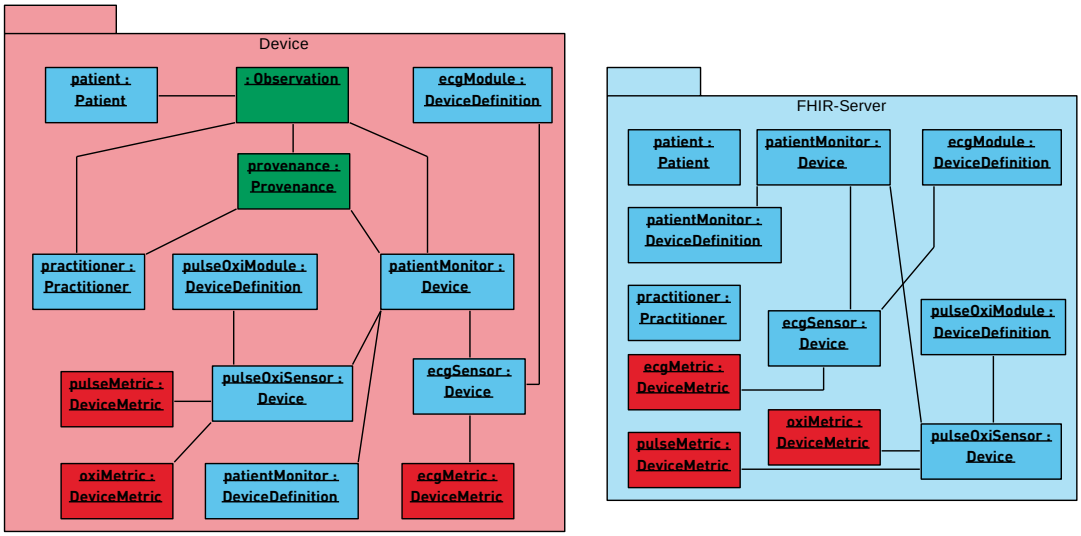

Figure 4: Object diagram: Distribution and origin of resources on device and server.

The not yet persistent devices are marked in green in the diagram and transferred from the device to the server. The resources marked in blue are created by other systems on the FHIR® server and are not changed or managed by the device. The device regularly receives updates of these resources from the server or requests them if necessary. The resources marked in red are resources that are created and managed by the device. In this example, these are the *DeviceMetric* resources, which include data from the calibration of the device. In this example, the *Device* resource is not managed by the device since it is assumed that the devices are controlled by appropriate personnel in management systems that transfer the current versions to the server.

3.4 Resource: Device

The *Device* resource is a management resource that tracks individual instances of active medical devices and their locations. Other resources use references to this resource to indicate which device has taken action, such as during an *Observation*, or which device was issued or prescribed for patient use. Besides, this resource type is used to capture and submit Unique Device Identifier (UDI) information about a particular device.

The FHIR® standard provides various attributes for the *Device* resource to allow adding contextual information. As part of the concept, the attributes that are of interest for security purposes were included in the class diagram in Figure 3. These include, for example, the serial number of the device, the location at which the device can be found, manufacturer information, and information on the availability of the device in the network. These attributes are further discussed in Section 4.

3.4.1 Device Identifiers

Almost all devices have one or more identifiers or codes assigned, printed, or attached to the device using barcodes or RFID tags. The identifier or code may come from the manufacturer, various institutes, and registers. Each of these identifiers assigned to the device can and should be recorded in the *Device* resource. The *identifier* attribute is for use only if it is an actual identifier for a particular instance of a device. That would mean that each device would have its serial number and would be represented with that element. *Device* and *DeviceDefinition* have the attribute *manufacturer* to specify information about the manufacturer.

3.4.2 Device Composition

A device can have reference to a *DeviceDefinition* resource. This resource describes the abstract type of a device. There is also a *DeviceMetric* resource, which describes a measurement function, calculation function, or setting function of a medical device (Health Level Seven International (HL7), 2018a).

The *Device* is the administrative resource for a particular real device, whereas *DeviceDefinition* and *DeviceMetric* model the physical part of a device. A *DeviceDefinition* represents a module of a device and allows the definition of hierarchical device configurations (devices as part of other devices) using the *parentDevice* attribute. The connection between *DeviceDefinition* and *Device* is made by the attribute *definition* of a *Device* resource. The composition of administrative devices is represented by the *Device*s pointing to their parent using the *parent* attribute.

Devices are designed as a composition of smaller devices, each containing their own *DeviceMetric* and either individual or shared *DeviceDefinition*s, to provide modularization of,e.g., measurement devices. The use of these resources is not regulated by FHIR®. Therefore, a device-specific mapping of a real device to the FHIR® resources is necessary.

3.5 Resource: Observation

Observation resources are a crucial element in healthcare and are used to support diagnosis, monitor progress, determine baselines and patterns, and even capture demographics.

The *Observation* resource either contains specific characteristics or is used to logically combine multiple *Observation* resources. The resource is referenced by *DiagnosticReport* to display laboratory, imaging, and other clinical and diagnostic data to produce a complete report. Each *Observation* represents a measured value. This complexity level is not considered in this paper since the procedure for handling *DiagnosticReport* resources is identical to that of *Observation* resources. Unlike the *Observation*, the *DiagnosticReport* resource typically includes an additional clinical context and a mix of atomic results, images, imaging reports, text, and coded interpretations, and formatted representations (Health Level Seven International (HL7), 2018c).

3.6 Resource: Provenance

The *Provenance* of a resource describes entities and processes and systems involved in the creation, provision, modification, or deletion of this resource. Therefore, it is an essential basis for assessing authenticity, building, and maintaining trust and reproducibility. The resources are created by the application that initiates the creation or update of the resource and transfers it with the other resources.

3.6.1 Targets, Subjects and Roles

The relationship between a resource and its *Provenance* is established by referencing the *Provenance* resource to its destination. The *target* attribute is used for this, which contains a reference to one or more targets. In this way, *Provenance* can be provided for any resource or version, including previous versions. There can be multiple *Provenance* resources for a particular resource or version.

The *agent* attribute embedded in the *Provenance* is an element that defines which systems, people, and devices were involved in which role in the creation or modification of a target resource. Thus, this element defines the attributes to show how the agent participated in the target resource change. The attribute *who* must be specified to express which subject has performed the specified roles. A reference to resources of existing entities *Practitioner*, *RelatedPerson*, *Patient*, *Device* or *Organization* is given. In the case of a *Practitioner* resource, further context information can be specified using a *PractitionerRole* resource (Health Level Seven International (HL7), 2018e).

It should be noted that *Practitioner* is used for all medical professionals. The attribute *code* with the coding system *PractitionerRole*[1] indicates in which specific medical role a *Practitioner* has acted. This role attribute makes sense, since doctors or nurses can have different roles in the clinical context. This coding system is kept open and includes many FHIR®-specific as well as existing and internationally used terminologies and nomenclatures (Health Level Seven International (HL7), 2018d). Within the example presented, this resource is not used to keep the complexity as low as possible. For practical use, however, it may be necessary to specify the role of the medical personnel as precisely as possible to meet legal requirements.

3.6.2 Electronic Signatures

The resource *Provenance* defines an attribute *signature*, which has the FHIR® data type *Signature*. This signature attribute is used to sign the *target* resource of a provenance digitally. A *Signature* resource contains an electronic representation of a signature and its supporting context in a FHIR® accessible form. The signature can be an electronic signature such as a XML signature or a JSON Web Signature (JWS). Further, it can be an image of a physical signature. There can be multiple signatures in one *Provenance*. Signatures for several referenced target resources and several signatures for one target resource can be attached. The latter can be useful if different subjects, such as people and

[1] *FHIR® coding system* PractitionerRole*:* *https://www.hl7.org/fhir/valueset-practitioner-role.html*

systems, were involved in creating a resource and want to attest to this participation and the correctness of the target resource. This is done using the FHIR® coding system *SignatureTypeCodes*[2] and the attribute *type*. The codes come from the American Society for Testing and Materials (ASTM) standard, E1762-95 (2013) *Standard Guide for Electronic Authentication of Health Care Information*, which regulates the use of electronic signatures in the context of Electronic Health Records (EHRs). However, in 2017 this standard was withdrawn due to a lack of use by industry.

The attributes *when* and *who* represent the machine-readable point when the signature was generated and the entity to which the key used for the signature is assigned. The attributes *targetFormat* and *sigFormat* define the MIME type of the referenced target resource and the MIME type of the signature according to Internet Engineering Task Force (IETF) Best Current Practices (BCP) 13[3]. These are for the *targetFormat* for FHIR® for resources in the XML representation *application/fhir+xml* and for JSON equivalent *application/fhir+json*. For the attribute *sigFormat* you can use common MIME types for images such as *image/png*, the electronic signature types *application/signature+xml* and *application/jose* for XML signatures and JWS[4]. The base64-encoded signature is then embedded in a *Signature* resource in the attribute *data* (Health Level Seven International (HL7), 2018h).

3.7 Resource: AuditEvent

All actors, systems, applications, processes, and services involved in an auditable event should record an *AuditEvent*. This results in multiple *AuditEvent* entries that indicate whether security measures, such as access controls, work properly. Therefore, it is common for an auditable event to be recorded both by the application in a process and by the involved servers. Security events are not limited to communication or RESTful events. These may also be,e.g., logins, access control decisions, or configuration changes. Servers that provide support for *AuditEvent* resources generally should not accept updates or deletions of resources because this would affect the integrity of the audit record.

A provenance resource contains overlapping information. It collections information about the context of the information change of the resource at the time of the event. *Provenance* resources are prepared by the application that initiates the creation or update of the resource and can be persisted at the destination resource.

3.8 JSON Web Signature (JWS)

JSON Web Signatures (JWSs) represent digital signatures of JSON objects, called payloads, as well as metadata about the signature. To achieve this, JWS defines the following so-called JWS claims:

[2] *FHIR® coding system* SignatureTypeCodes*:* *https://www.hl7.org/fhir/valueset-signature-type.html*
[3] *IETF BCP 13 - Multipurpose Internet Mail Extensions (MIME) Part Four:* *https://tools.ietf.org/html/bcp13*
[4] *RFC 7515 - JSON Web Signature (JWS):* *https://tools.ietf.org/html/rfc7515*

- ***alg* (Algorithm):** The *alg* (algorithm) header parameter specifies the algorithm used to create the signature. The signature is considered invalid if the implementation does not understand or support the specified algorithm. This header parameter must be present.
- ***jwk* (JSON Web Key):** The *jwk* header parameter is the public key that belongs to the private key used to sign the JWS. This key is represented as a JSON Web Key (JWK).
- ***jku* (JWK Set URL):** The header parameter *jku* (JWK Set URL) is an URI that refers to the JWK used for the signature. This key is represented as a JSON Web Key (JWK).
- ***kid* (Key ID):** The header parameter *kid* (key ID) indicates which key was used to create the JWS. With this parameter, senders can explicitly signal a change in the key for recipients. The structure of the *kid* value is not specified. When used with a JWK, the *kid* value is used to verify that the correct JSON Web Key (JWK) is used.
- ***x5u* (X.509 URL):** The header parameter *x5u* is an URI that references a resource for the X.509 public-key certificate or certificate chain that corresponds to the key used to sign the JWS. The identified resource must provide a representation of the certificate or certificate chain that is in PEM-encoded form.
- ***x5c* (X.509 Certificate Chain):** The header parameter *x5c* (X.509 certificate chain) contains the X.509 public-key certificate or certificate chain that corresponds to the key used to sign the JWS. The certificate or certificate chain is represented as a JavaScript Object Notation (JSON) array of certificates. Each string in the array is a base64-encoded PKIX certificate in DER representation.
- ***x5t* (X.509 Certificate SHA-1 Thumbprint):** The header parameter *x5t* (SHA-1 fingerprint X.509 certificate) is a base64-encoded SHA-1 fingerprint that uses the DER encoding of the X.509 certificate as the one used for the digital signature Key corresponds.
- ***x5t#S256* (X.509 Certificate SHA-256 Thumbprint):** The header parameter *x5t* is equivalent to the parameter *x5t*. Instead of SHA-1, the SHA-256 algorithm is used for the fingerprint of the X.509 certificate.

4 Results

This section elucidates the analysis of the FHIR® principles and resources concerning the preservation of resource integrity. It will outline the requirements and recommendations that need to be implemented to achieve the discussed security goals. These requirements are not defined by the standard itself.

4.1 Modeling a Patient Monitor

For the analysis, the preceding Section 3.4.2 demonstrates the composition of a patient monitor. The first step is modeling the patient monitor for which three devices, three metrics, and three definitions are necessary, as shown in Figure 2. Only the metric for the pulse measurement is considered for the scenario to reduce complexity.

The device is divided into two submodules for measuring ECG and pulse oximetry values. These modules and the patient monitor itself each have a *DeviceDefinition*. The monitor has a name as an example attribute, an *identifier*, an *id*, as well as a version in the *meta* attribute. It can be referenced via these attributes and elements, as discussed in Section 3.1. The patient monitor is perceived as one single device but consists of a composition of different measurement systems that can be represented as separate devices in FHIR®, as presented in Section 3.4.2. This composition is particularly useful because these subdevices may be encapsulated on different hardware platforms. These individual modules would be modeled with FHIR® as their own devices, which have a reference to the actual device. An pulse oximetry module object is shown in Figure 5. Similar to the patient monitor, it contains all attributes required for identification. Besides, it references the patient monitor with its identifying characteristics in the *parent* device attribute.

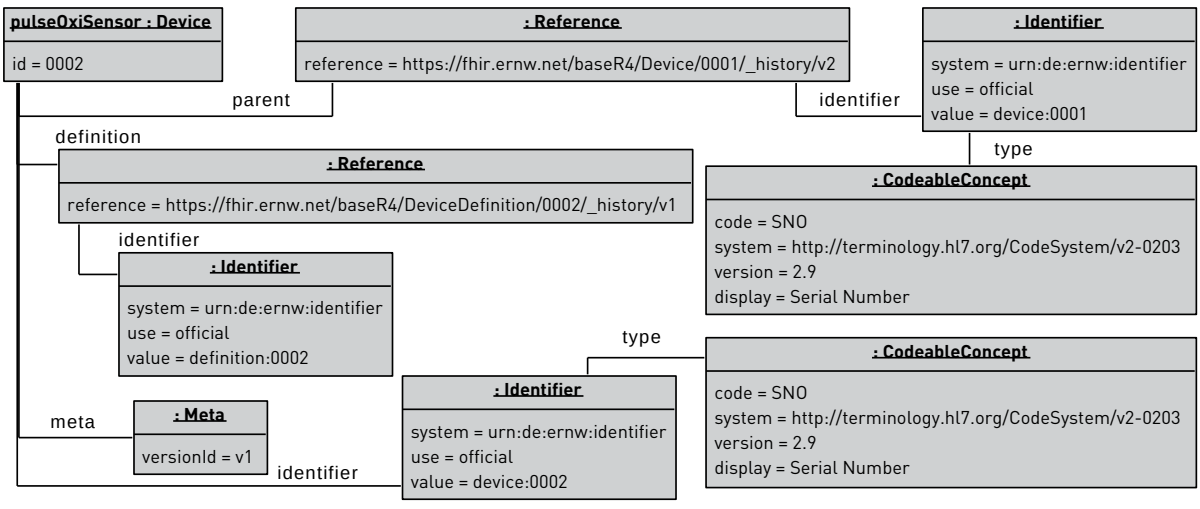

Figure 5: Object diagram of the pulse oximetry sensor.

Each module can have a *DeviceDefinition* that describes the type of module and contains general information. The definition of the pulse oximetry module is shown in Figure 6. This definition is kept to a minimum in the example. Still, in real applications, further information,e.g., safety instructions, contact details of competent authorities, and the module's manufacturer and functionality descriptions, may be added.

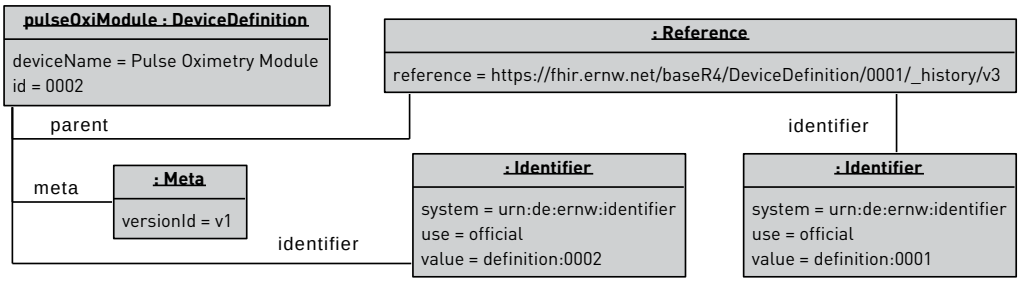

Figure 6: The object diagram shows the definition of the pulse oximetry definition.

The *DeviceMetric* resources associated with the module define the capabilities of the individual measurement system, such as an electrocardiogram (ECG), pulse oximetry, or oxygen saturation. For a 12-lead ECG, twelve *DeviceMetric* resources would be modeled. In this example, only one metric is modeled and represented. For the pulse oximetry module, own metrics are defined for the pulse and the oxygen saturation. The metric for pulse measurements is shown in Figure 7.

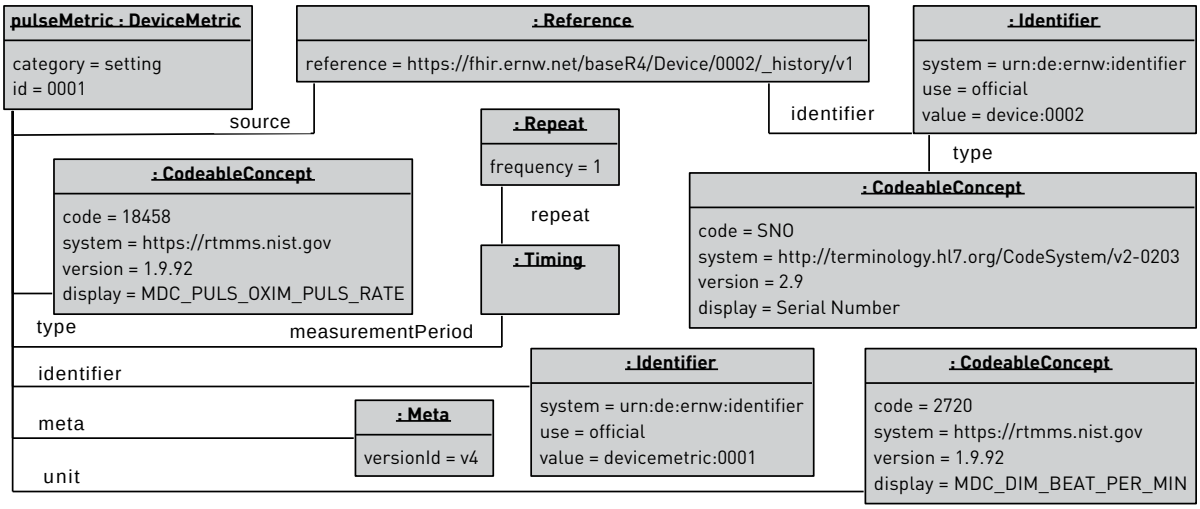

Figure 7: The object diagram shows the definition of the pulse metric.

In addition to the identifying characteristics, the metric defines the reference to the relevant device resource and measurement specifications. The metrological information includes the *measurementPeriod*, which represents the fre-

quency at which the device generates measured values. The unit is implicitly specified by FHIR® and always *number per second*. On the other hand, the measurement-related information includes a machine-readable, coded specification of the unit's measurement values and coding of the measurement value. This machine-readable coding enables automatic semantic validation and evaluation of the measured values of various devices.

Figure 8 shows how the *observation* in our example represents a measured value of the patient monitor. Please note that an observation does not have a logical ID if a device creates it since this is only assigned by a FHIR® server. However, the resource shown contains a business identifier. The resource references the patient to provide context information about the medical personnel who carried out the measurement and the device used for measuring. In addition to this information, the meaning of the measured value itself is coded in a machine-readable format in the element *code*. The specific expression in the element *value* still encodes the unit of the measured value.

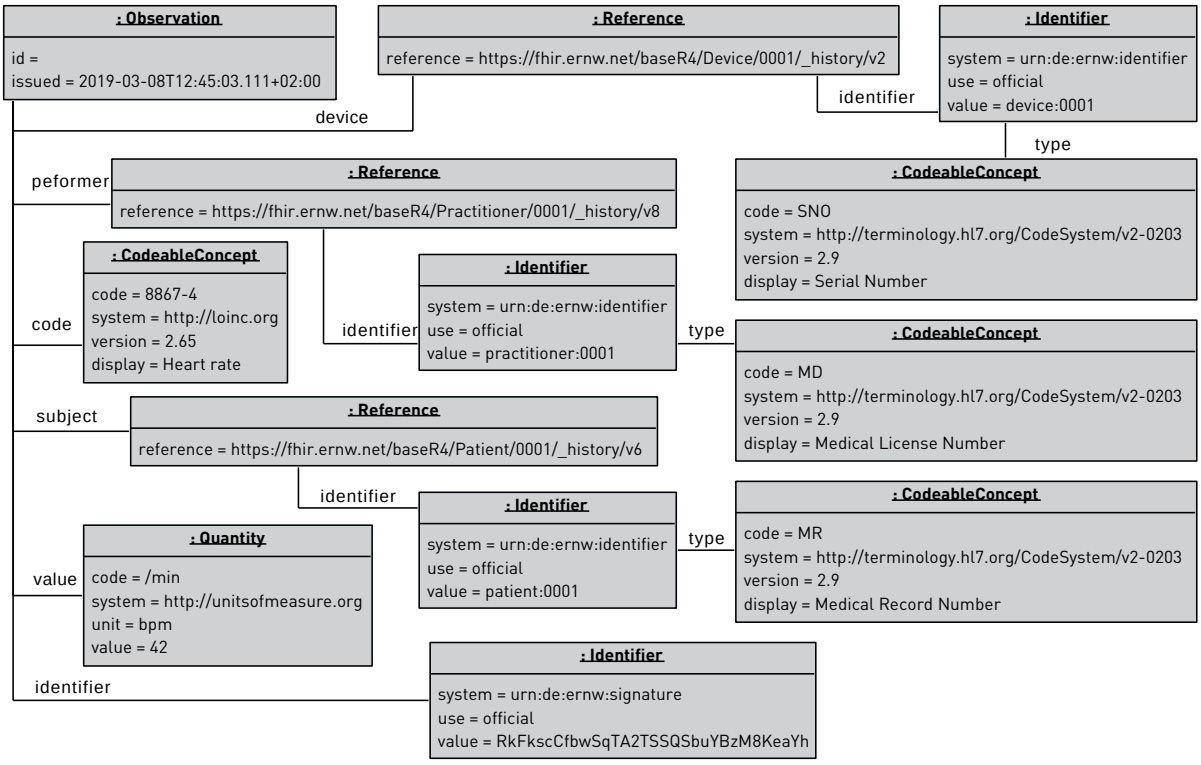

Figure 8: Object diagram: Observation as realization of a patient monitor's measurement results.

4.2 Accountability Requirements

Accountability enables a change to be attributed to a subject. Measures to ensure accountabilities are the audit-proof logging of all actions and the use of digital signatures as well as timestamps. A *Provenance* that is supposed to state

context for a resource can involve several entities, such as people and systems, in its creation or modification. This *Provenance* may have multiple signatures. Provenances can also refer to one or more resources. As part of the concept, references with version numbers must always be used for these references so that the reference to a specific instance becomes clear. Whenever resources are changed, provenances must be created, signed by the device, or the human editor. The *recorded* attribute defines the time of the creation of the *Provenance* resource. From a traceability perspective, this point in time must reflect when the target resources are changed, not the time the *Provenance* is transferred to the server. A signature created using JWS means that URLs, identifiers, and internal references are frozen and cannot be changed. This may be a desirable feature, but it can also affect interoperability between closed ecosystems, in which resources often are re-identified.

4.3 Identification of Affected Resources in the Event of a Compromise

To log the creation, modification, and deletion of resources and the entities involved, the FHIR® standard provides for the use of *Provenance* resources (Health Level Seven International (HL7), 2018e). As described in Section 3.6, the *target* attribute is used to provide references to resources affected by creation, modification or deletion. Further, the *agent* attribute shall be used to name systems, people, and devices involved in the creation, modification, or deletion of the referenced resources. The *recorded* attribute indicates when the activity using the *Provenance* was documented. To identify all resources in an incident response case being created, modified, or deleted by a compromised system or device, all *Provenance agent* attributes must be queried for the systems under investigation. In this way, affected resources can be identified using the provenances' *target* attribute. To ensure that all potentially untrustworthy resources are identified with this approach, an FHIR® implementation must provide that the requirements set out in the following and previous subsections are met.

4.4 HTTP Request Handling

To ensure that every change to a resource is documented by a *Provenance*, each FHIR® server must ensure that corresponding HTTP requests are only accepted with respective provenances. The *target* attribute of the *Provenance* must contain a reference to the corresponding resource. Further, for every resource, at least one provenance must exist. Additionally, each *Provenance* must reference all actors involved in the *agent* attribute. In the case of the discussed patient monitor, this may refer to the operator and the device. Further, every FHIR® server must ensure that the associated provenance contains a valid signature from every actor referenced in the *agent* attribute.

4.4.1 Creating Resources

There are two defined alternatives in the FHIR® standard for creating new resources, creating interactions via HTTP POST, and updating interactions via HTTP PUT.

4.4.1.1 Creating Resources Using HTTP POST

With the *create* interaction, the client sends an HTTP *POST* request to a FHIR® server. The HTTP message body must contain a FHIR® resource, whose *id* attribute can be empty. To conform to the standard, the server must fill in the *id* attribute and the metadata elements *versionId* and *lastUpdated* and ignore any values sent for these elements. If the FHIR® server accepts the resource, it replies with a location header, the assigned logical ID, and the resource version, which is *0* for newly created resources.

The difficulty for the scenario in question is that the shared provenance cannot yet contain the created resource's logical ID and version. If a placeholder is used and the server modifies the *id* attribute, the metadata elements *versionId* and *lastUpdated* of the created resource as well as corresponding changes to the *target* attribute the provenance would break the resource's signature as part of the provenance.

4.4.1.2 Updating Resources Using HTTP PUT

The second way to create resources is by using update operations. The client sends an HTTP *PUT* request to the FHIR® server. In this way, the client can determine the resource's logical ID if it does not already exist. It must be ensured that the newly created resource does not overwrite an existing resource.

The use of conditional updates, which are still in a test phase in the FHIR® standard, could pose a solution. Interacting with conditional updates allows a client to update an existing resource based on some identification criteria instead of updating it based on logical IDs, which in the present case, is still unknown.

When processing the request, the server performs a search on the resource type to resolve a single logical ID. The action depends on how many results are found, as the following list shows:

- No matches, no ID specified: The server creates the resource.
- No matches, ID specified: The server treats the interaction as create via update.
- A match, no ID specified or ID specified, and this matches the resource found: The server carries out the update against the appropriate resource.
- A match, ID specified, but this does not match the resource found: The server returns a *400 Bad Request* error.
- Multiple matches: The server returns a *412 Precondition Failed* error.

4.4.2 Updating Resources

As described in the previous section, HTTP *PUT* requests are used to initiate an update of resources. The request's body must represent a resource with an ID identical to the ID specified in the URL. The FHIR® standard specifies that the server should accept the transmitted resource as it is transmitted except for the elements *.meta.versionId* and

.meta.lastUpdated, as described in Section 4.5. This is important; otherwise, the resource's signature as part of the associated Provenance could no longer be validated.

4.4.3 Deleting Resources

As medical data often has to be kept in the healthcare system for a long time, not only because of regulatory requirements, a *DELETE* request must not delete the resource. Whether the *DELETE* method is supported at all for certain resource types or a specific instance is at the discretion of the server. For the resource types *Provenance* and *AuditEvent*, this must not be allowed.

As described in 4.5, FHIR® servers that store resources must keep a version history for resources. From a version history perspective, deleting a resource is equivalent to creating a particular type of history entry with no content where the resource is marked as deleted. Therefore, the current version of a resource is an empty entry in the version history marked as deleted. However, all previous versions of the resource still exist on the server because the version history must not be deleted from the server. Consequently, a resource deletion can be undone by updating the resource described in 4.4.2. The Handling of version conflicts is possible by sending an HTTP *ETag* header with a *DELETE* request's response. A resource whose current version is an entry marked as deleted can no longer be read or searched by clients using *GET* requests.

4.4.4 Searching and Requesting Resources

If a client sends a *GET* request specifying the resource type and the logical resource ID, the server replies with an HTTP status code *200* and the *current* version of this resource assuming the resource exists and the client is authorized to access the resource. To search for resources that meet specific search criteria, the client sends a *GET* request to the resource type according to the following syntax, where parameters describe the search criteria.

```
GET [base]/[type]{?[parameters]{&_format=[mime-type]}}
```

The client cannot specify any attributes of the resource type for a valid request as search criteria, but only those that the server has implemented. The server responds with a HTTP status code *200* and a *bundle* of the type *Searchset*, which contains the results of the search as a collection of zero or more resources in a defined order.

The authorization concept of an FHIR® environment must ensure that the *bundle* only contains resources that the client can access.

4.5 Version History

In environments where several systems communicate directly with each other and request and change resources, version management of these resources is essential to avoid so-called *Lost Updates*. Besides, version history and thus the

possibility to access all old versions of a resource is necessary to roll back the potentially corrupted data in the event of a compromise. In the FHIR® standard, resources are represented as a logical unit by a series of versions, one of which is marked as *current* version.

The creation of resources or updates and the deletion of existing resources create new versions of a resource. This also applies if these are triggered by processes that do not use the RESTful API. The FHIR® standard supports versioning with the help of two metadata elements that each saved resource must contain, *.meta.versionId* and *.meta.lastUpdated*. These metadata elements correspond to the HTTP headers *ETag* and *Last-Modified*. According to the FHIR® specification, these two elements are always updated by the server when the corresponding resource is changed. The server ignores client-specified values for these items. The version ID of a resource does not have to be changed incrementally. It only has to be unique for each version of a resource with a specific resource ID.

FHIR® servers must implement the *History* service described in the FHIR® standard at the level of the individual resource instances. If the client has the necessary permissions, the server replies with a *Bundle* of the type *History*, which contains the requested version history, sorted according to the age of the versions and including deleted resources. If a system has been identified to be compromised within a FHIR® environment and the search of all *Provenance* identifies each resource that was created, modified, or deleted by a compromised system, the resource ID can be used to request the version history of the affected resources. A *Provenance* must always reference the version of a resource. In this way, it is possible to understand which changes to a particular resource are due to the compromised system.

4.6 Creating AuditEvents

A FHIR® server must create an *AuditEvent* instance when processing a *GET* request. An *AuditEvent* must specify one or more agents that are part of the event that caused the creation of the *AuditEvent*. The event must include at least the requesting client and the replying server. Furthermore, the *AuditEvent* must contain all resources sent to the client identified by their logical ID and version.

Moreover, servers that store *AuditEvent* must be able to search for events by entities and actors. In this way, entities having requested and processed resources from a compromised device can be identified. This allows inferring direct or indirect impact on diagnoses and therapy recommendations.

However, if provenances prove that potentially compromised data may have been processed, all subsequent decisions should be subjected to a review.

4.7 Electronic Signatures

This section deals in detail with the implementation of electronic signatures. The use of X.509 certificates is mandatory. These certificates enable an explicit binding of keys to validated identities and are the only means to allow an electronic

and automated validation of identities and *Provenance* signatures. In the following, the term *key* or *public key* will be used to discuss requirements for the environment, assuming that the keys are bound to and may only be retrieved with a respective certificate.

4.7.1 Identification of Keys

The recipient of a JWS must determine the public key matching the private key used for the digital signature. This is only possible using X.509 certificates. The used public key can be identified with the header parameters described in Section 3.8. The header parameters' integrity must be protected to prevent any signatures from being created and treated as trustworthy. The producer should include sufficient information in the header parameters to identify the public key unless it uses other means or conventions to determine the key used. The key's ID may not be sufficient as the signing identity may also be of interest. Therefore, a reference to the identity's certificate should be added using *JWK* headers such as *x5c* or *x5u*. Additionally, the certificate fingerprint shall be added using,e.g., the *x5t* and *x5t#S256* headers. The signature check must fail if the key or the key's identity cannot be determined.

4.7.2 Requirements for the Secure Transfer of Public Keys

Implementations that support the header parameters *jku* or *x5u* and can retrieve public keys from external sources must transfer these keys using transport-layer protected channels such as HTTPS. The confidentiality, authenticity of the external source, and integrity of the information retrieved must be strictly verified.

4.7.3 Key Management

The protection of the keys is fundamental. By compromising the private key of a signing entity, an attacker could impersonate this entity. Further, the technique used to retrieve public keys must authenticate the origin of the key. Otherwise, it cannot be determined which entity signed the resource.

4.7.4 JSON Security

A strict JSON validation must be present before validating the signature. Ambiguous situations can arise if the JSON parser does not reject incorrect JSON syntax. This includes that the header parameter must be unique, i.e., a *JWS* with duplicate header parameters must be rejected.

4.7.5 Detached Content

With *JWS*, it is possible to protect the integrity of content that is itself not contained in a *JWS*. One way to achieve this is to create a *JWS* with a representation of the content as a payload, deleting the payload from the *JWS*, and transmitting the changed *JWS*. This method assumes that the recipient can restore the exact payload used in the *JWS*. FHIR® enables

the use of this kind of signature since the signed resource is directly accessible within every request or message. During validation, the recipient reconstructs the *JWS*. Therefore, the missing payload is hashed with the header. This hash is compared to the hash returned from decrypting the signature with the signer's public key.

Detached signatures simplify the certificate validation as the payload of an attached signature corresponds to the resource's duplication. When attached signatures are required, the signature's payload and the target resource must be checked for equality. Otherwise, attackers may be able to reuse existing valid signatures, including payload.

4.8 FHIR® Limitations

This section lays out the limitations identified during the analysis.

4.8.1 Missing Target Attribute

As of FHIR® version 4.0.1, no *target* attribute is present for *Signature* resources. As a *Provenance* may contain multiple *Signature* instances for one and more target resources, it remains unclear which payload was signed. This complicates the signature verification as in the worst case all possible target resources need to be tried to identify the correct one. Further, a track of all resources and the matching signatures must be present to identify resources without signatures. This computational effort needs to be performed every time the signatures are checked. As a consequence, it is recommended to add a *target* attribute with the resource type *Reference* to the *Signature* resource to enable a unambiguous signature validation.

4.8.2 Signing Provenances

It may also be noted that the FHIR® standard states that the *target* attribute should not be added to the *Provenance*[5]. The standard states that the server will determine the respective resource ids in the HTTP request body and fill this field. We recommend adding these fields as it may also be desired to add a signature for the *Provenance* itself. In this case, the signature may be calculated on the *Provenance*, including the *target* attribute and all other signatures. Afterward, the provenance's signature may be transmitted either using a custom HTTP headers such as *X-Provenance-Signature* or by including the signature in the *Provenance* itself. The last possibility enables us to verify the signature after transmission, too, as the signature may be persisted with the other signatures. This requires the *Signature* resource to have a target attribute and the *Provenance* instance to have an identifier.

4.8.3 Assigning Identifiers

Even with the *Signature* resource having a *target* attribute, the reference to the resources cannot be constructed, as the resources may not have a logical ID yet. Therefore, it is needed to generate a business identifier for the resource. As

[5] *https://www.hl7.org/fhir/provenance.html, accessed: May 12, 2020.*

described in Section 3.4.1, the business identifier, which is stored in the *identifier* attribute and has the resource type *Identifier*, is uniquely coded using public coding systems. A dedicated signature-only coding system may be used to create a signature when no other identifier is present yet. The requirement for the identifier's uniqueness is that it is at least unique for the resources transferred with the provenance. Though, a random generation with enough entropy is advisable to avoid collisions. As an example, the resource's SHA-256 hash may be used appended to the creation timestamp.

5 Discussions

This section is opened with a summary of this paper. Afterward, a discussion of the results and analysis is carried out. Finally, a short outlook is given.

5.1 Summary

This work focussed on the evaluation of preserving resource integrity and accountability in distributed FHIR® communication systems.

The FHIR® resources *Provenance* and *Signature* were essential for the work. Functionalities offered by FHIR®, as well as mechanisms for maintaining the security of a FHIR® system, were analyzed. An exemplary patient monitor was used to show how the methods and resources provided by FHIR® can be used to preserve resource integrity and which conditions for the behavior of a FHIR® server and its system environment need to be met.

This work shows that the FHIR® standard has inconsistencies, particularly in the area of electronic signatures. The *Signature* resource shall be modified to have a `target` attribute enabling signatures to reference signed resources.

5.2 Discussion

This work shows that the communication standard FHIR® offers design possibilities to achieve the previously discussed protection goals. However, the requirements and recommendations concerning electronic signatures are not yet precise enough or elective.

An example is that for electronic signatures with JWT, claims are defined, and their use must be specified and required in the standard. Signing resources using JWS results in URLs, identifiers, and internal references being frozen. This may be a desirable feature, but it can also affect interoperability between closed ecosystems, in which resources are often re-identified.

Also, if the results of this work are not generally applicable but are limited to the specific example, it was shown which problems currently exist in implementing designed security requirements in FHIR®. One example is that FHIR® signatures do not currently specify which resource was signed. There are also gaps in identifying resources for signing before the server's resources receive a logical ID and version. A partial solution for this was shown by using a signature-specific identifier generated by the client at random. This identifier is only used for the initial creation of the resource on the server.

Another result of this work is that a device must communicate changes to its settings, updates, or status to ensure the database's traceability and consistency. For this purpose, update requests need to be appropriately secured.

In this work, requirements for the environment of a FHIR® system were set out. It should be noted here that these requirements in the clinical environment entail a significant operational effort, which is clear from the example of the operation of public key infrastructures. A particular implementation of this environment, its configuration, and the security concept were not part of this work.

Another limitation of this work is that only RESTful endpoints of a server were considered, but not the messaging API, documents, or service-oriented architecture.

It remains open to how a rollback of data from compromised devices is possible without losing changes to resources made during the period of compromise by other systems. Nevertheless, it was shown how these devices could be reliably identified by preserving resource identity.

5.3 Outlook & Future Work

It remains open to which extent subsequent versions of the FHIR® standard will evolve in terms of security. For this purpose, as many different scenarios and medical processes as possible in interaction with medical devices, systems, and personnel should be evaluated to make statements as specific as possible. These statements should be aligned with the FHIR® principles.

6 References

Bedner, M, & Ackermann, T. (2010). Schutzziele der IT-Sicherheit. *Datenschutz und Datensicherheit - DuD*, *34*(5), 323–328.

Bundesamt für Sicherheit in der Informationstechnik (BSI). (n.d.). *IT-Grundschutz-Kompendium - Glossar*. Retrieved October 19, 2020, from https://www.bsi.bund.de/DE/Themen/ITGrundschutz/ITGrundschutzKompendium/vorkapitel/Glossar_.html

Health Level Seven International (HL7). (2018a). *Device - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/device.html

Health Level Seven International (HL7). (2018b). *Identifier - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/datatypes.html#identifier

Health Level Seven International (HL7). (2018c, December 27). *Observation - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/observation.html

Health Level Seven International (HL7). (2018d). *PractitionerRole - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/practitionerrole.html

Health Level Seven International (HL7). (2018e). *Provenance - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/provenance.html

Health Level Seven International (HL7). (2018f). *Reference - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/references.html

Health Level Seven International (HL7). (2018g). *Resource - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/resource.html

Health Level Seven International (HL7). (2018h). *Signature - FHIR v4.0.1*. Retrieved October 19, 2020, from https://www.hl7.org/fhir/datatypes.html#Signature

Miller, M. (2008). *Modeling the trust boundaries created by securable objects* (LS Group, Ed.). Retrieved October 19, 2020, from http://hick.org/~mmiller/presentations/woot08/trust-boundaries-sec-objects.pdf

MITRE. (n.d.). *CWE-501: Trust Boundary Violation*. Retrieved October 19, 2020, from http://cwe.mitre.org/data/definitions/501.html

Shostack, A. (2014). *Threat Modeling: Designing for Security*. Wiley Publishing.