TM Forum Technical Report

Security Ontology

TR291I

Maturity Level: **General Availability (GA)**	Team Approved Date: 14-Mar-2025
Release Status: Pre-production	Approval Status: TM Forum Approved
Version 3.7.0	IPR Mode: RAND

Notice

Table of Contents

Executive Summary

The ontology model defined in this document defines the necessary vocabulary that can be used in the security domain. The focus of this document is the specification of how a security intent is expressed.

This document provides an ontology and vocabulary for definition and formal description of security within the TM Forum Intent Ontology (TIO).

1. Introduction

The complexity of managing a mobile network such as 5G and future networks like 6G has kept increasing due to the various subsystems, technologies, and services that must be supported. Additionally, there are new emerging and evolving attacks on different computing stacks, that are getting more complex, distributed, and targeted. To ensure smooth and resilient operation, scalability, proper configuration of the multitude of settings, policies, and service parameters, including those required for secure operations, is required. Adopting zero-touch and autonomous capabilities (i.e self-protection capabilities) to security will support facilitating a more risk-based decision-making for security teams, simplify management, automate deployments and enhance security KPIs, for example reducing Mean Time To Detect (MTTD) and Mean Time to Response (MTTR), and reduce human intervention to ensure faster automation loops to provision Control, and detect, respond and prevent against threats.

Intent-based security management is an enabler to achieve more simplified automation for security. The usefulness of intent for security revolves on two main aspects, first the *abstraction* level used and presented to the end user in the form of requirements to be measured and fulfilled, and in the way, requirements are further modeled and described using ontologies. These aspects of abstraction enable to move towards security solutions that are model-driven, where the abstraction of the security concepts and processes will be a simplification for the security teams. The second useful aspect, is the *underlying closed loop* that is composed of different agents that are responsible on monitoring, evaluating, planning, and actuation. Designing and implementing security automation loops that ensure fast analysis and decisions and at the same time ensure that the decision to be actuated is not intrusive and will not impact the system KPIs is a powerful capability that will enhance the security capabilities with better detection times.

1.1. Scope

The TM Forum Security Ontology is designed for expressing security intents as part of intent driven systems and control loops. This document introduces classes and properties for defining and describing security.

1.2. Revision Information

This revision v3.7.0 of this document is the first version of the Security Ontology model. It is aligned with the TIO version 3.6.0. The document is part of the TMF Forum Intent Ontology v3.

2. Notation and Dependencies

The security ontology model is part of the TM Forum Intent Ontology and depends on the following models:

Model	Prefix	Namespace	Published by	Purpose in the model
Security Ontology	sec	`http://tio.models.tmforum.org/tio/v3.6.0/SecurityOntology`	TM Forum	Specifications of security (This document)
Function Definition Ontology	fun	`http://tio.models.tmforum.org/tio/v3.6.0/FunctionOntology`	TM Forum	Model for formal description of functions
Intent Common Model	icm	`http://tio.models.tmforum.org/tio/v3.6.0/IntentCommonModel/`	TM Forum	Defines basic vocabulary and concepts of intent based operation.
RDF version 1.1	rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	W3C	Providing fundamental modeling artifacts
RDF Schema 1.1	rdfs	`http://www.w3.org/2000/01/rdf-schema#`	W3C	Providing fundamental modeling artifacts
XML Schema	xsd	`http://www.w3.org/2001/XMLSchema#`	W3C	Providing of data types for literal objects
Examples	ex:	`http://www..example.org/`	IANA	Reserved domain name for examples

Table 1.1: Model references

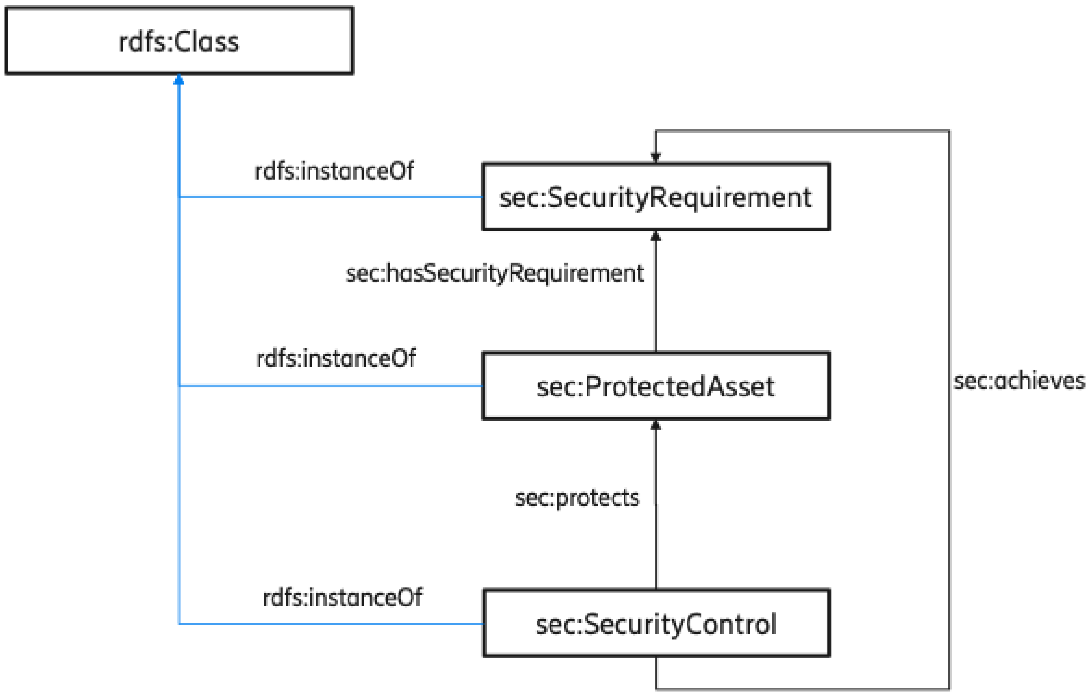
The Security Ontology model is based on the Resource Description Framework (RDF) [rdf, rdf_mt, rdf_primer] and the Resource Description Framework Schema (RDFS) [rdfs] published by the World Wide Web Consortium (W3C).

2.1. Notation and Conventions

The examples provided in this document are presented using the terse RDF triple language (TURTLE) syntax [turtle]. For referring to objects defined in other models and namespaces, the examples use prefixes as defined in Table 1.1. without introducing them with explicit statements in every example. Additionally, the generic namespace prefix "`ex:`" is used for defining example objects. Examples also use XML Schema with the prefix "`xsd:`"

3. Security Domain

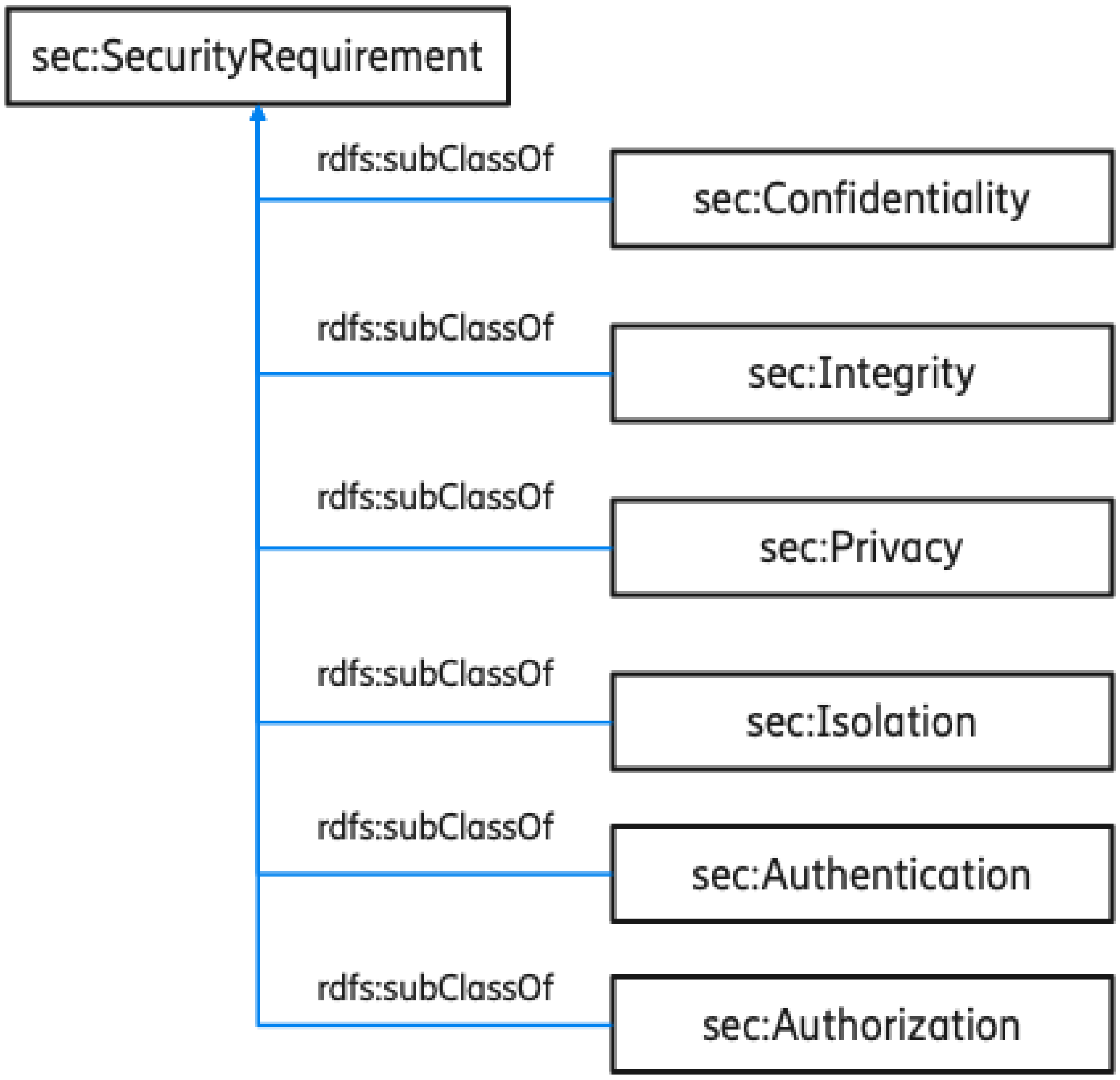
Security can be described as a condition or a state that results from the protective measures that enable an asset to perform its functions despite risks posed by threats, thus a secure state is achieved by preserving different attributes or properties of a system. These properties are usually described in terms of confidentiality, integrity and availability. They can be extended to include other properties as well, such as privacy, authentication or any other property that if is not fulfilled by the system it will induce harm and loss to the system. Achieving a secure state of an asset or a network is a result of understanding the asset that requires protection and is a target of an attacker, and what are the properties necessary to ensure the asset's security, where preserving the properties is facilitated by leveraging one or more Control.



Hence, the security domain can be described as the relationship between the asset that has a set of security requirements that are translated to the properties that are required to be preserved and the set of security Control that will be used to protect the asset. A continuous adapted and automated understanding of these relationships will reduce the risk of threats and ensure the reliability of the asset functionality. In the TIO intent ontology, security is described using `sec:SecurityRequirement` which represents the properties or attributes in a system that are required, the `sec:ProtectedAsset` is used to define the asset that requires protection, and `sec:SecurityControl` presents the Control that can be used to protect the Asset and achieve the security properties or requirements

4. Security Requirement

A security requirement in the TM Forum intent ontology is expressed by instances of class `sec:SecurityRequirement` . The main requirements in the security domain refer to confidentiality, integrity, authentication, and additionally isolation and privacy. The `sec:SecurityRequirement` class has five subclasses, each subclass represents a security requirement `sec:Confidentiality`, `sec:Integrity`, `sec:Authentication`, `sec:Isolation` and `sec:Privacy`.



The `sec:SecurityRequirement` class has set of requirements that are defined that can either be used with the protected asset to define the security requirements needed by an asset `sec:hasSecurityRequirement` , or they can be used to infer which security control can provide or achieve the security requirement by using the property `sec:achieves`. There is also a backward relationship between `sec:SecurityRequirement` and `sec:SecurityControl` by using `sec:achievedBy` . For example, a subscriber identifier in a telecom network will have a privacy requirement `sec:hasSecurityRequirement sec:Privacy`, and an encryption control can achieve a confidentiality property `sec:achieves sec:Confidentiality`. A security requirement is leveraged to infer the suitable security control from the ontology that can be used to satisfy the requirement.

The security requirement is used as part of an instance of a network function, where the security requirement is related to a specific component (i.e asset) within the network function that needs to be protected. The below example shows an instance of a gNB that has user plane backhaul interface instance that has a security requirement property `sec:Isolation`.

```
ex:UPBackhaul01
   a sec:ProtectedInterface;
   rdf:type ex:BackhaulInterface;
   ex:hasStack ex:UserPlane;
   sec:hasSecurityRequirement sec:Isolation
.
ex:IsolationControl   a sec:SecurityControl;
   sec:achieves sec:Isolation
.
ex:SecureBackhaulgNB01
  a ex:NetworkFunction;
  rdf:type ex:gNB;
  ex:hasInterface ex:UPBackhaul01;
  sec:hasSecurityControl ex:IsolationControl
.
ex:T1
  a icm:Target; set:intersection ( [ set:resourcesofType (ex:gNB) ]
[ set:resourcesWithPropertyObject ( ex:hasLocationArea "TA#1337" ) ] )
.

ex:SecurityControlTarget
  a icm:Target;
  rdfs:member ex:IsolationControl
.

 ex:E1
  a icm:DeliveryExpectation;
  icm:target ex:T1
.

ex:E2
  a icm:DeliveryExpectation;
  icm:target ex:SecurityControlTarget
.
```

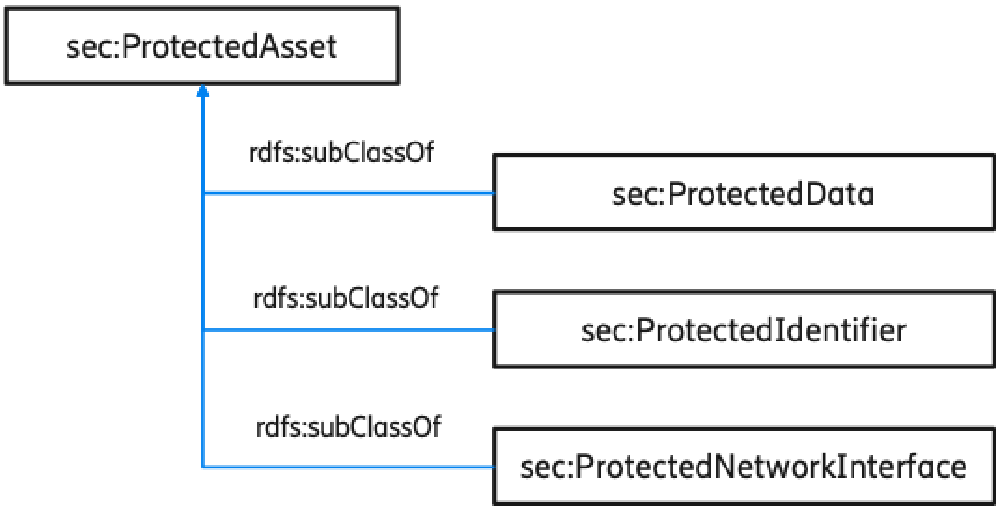
This example shows that the target `ex:T1` is defined to be the `set:intersection` of the resources of type gNB and a location property that can be used to identify geographically the targets that will fulfill the expectations. `ex:IsolationControl` is an instance of `sec:SecurityControl` that is defined as a control that will achieve the isolation requirement. The instance of the gNB uses a property `sec:hasSecurityControl` to present the relationship between the gNB that will be delivered and the security Control that will be delivered with it to achieve the isolation requirement. `ex:IsolationControl` instance is yet another functional requirement that is a part of the target `sec:SecurityControlTarget` . This example shows two delivery expectations where one has the set of gNBs as one target and the other delivery expectation has the security control as the other target.

5. Protected Asset

In security, it is necessary to have an understanding on what is the asset that is required to be secure and what are the security requirements the asset follows and should be fulfilled. An asset represents the items or areas of interest in a network or a system that needs to be protected, for example an asset can be the whole network function, for example gNB or a sub-part of an asset such as a subscriber identifier, for example SUPI or a network interface for example N3 or even a specific protocol on an interface that can have its own security requirements that is separated from other protocols, for example in the case of Radio Interface, a protocol like Radio Resource Control (RRC) can have its own requirements like confidentiality or integrity.

An asset that has security requirements in the TM Forum intent ontology is expressed by instances of class `sec:ProtectedAsset.` Nevertheless, on the same network node there could be different requirements for the node's subcomponents for example different Interface, Identifier or functionalities, that will require different protection mechanisms. The definitions of the protected Asset is leveraged to infer the suitable security Control for each asset to be protected, hence a network node is not considered an asset to be protected, it will rather have relationships with the sub-parts (i.e sub-Asset) that are part of a network node.

Since each asset can have its own set of properties and relevant descriptions, the different Asset are defined in this model as `rdfs:subClassOf sec:ProtectedAsset`.



The `sec:ProtectedAsset` class has three subclasses

- `sec:ProtectedData:` where an instance of this subclass specifies the data that has a security requirement.
- `sec: ProtectedIdentifier` : where an instance of this subclass specifies an identifier that is being used by a subscriber or a machine that has security requirement.
- `sec:ProtectedNetworkInterface` : where an instance of this subclass specifies the interface on the network function that has a security requirement.

The specification of each asset can be represented in other ontologies where it can be leveraged by the security ontology.

5.1. Protected Data

Data on a network function is a crucial asset that an attacker is usually trying to compromise. The data can be anything ranging from software binaries, machine learning models, training data, secrets or any other type of information that is deemed crucial and damage or compromise of the data can lead to the damage to the functionality of a network function.

The `sec:ProtectedData` is a subclass of `sec:ProtectedAsset`. For example, a ML model that is implemented on the network function can be treated as data of high value that requires confidentiality as a security requirement, or secrets that are used as tokens for machine to machine communication and stored on volumes on the network function might also require protection to protect it from being leaked in case the network function is compromised.

The different categories of data might require different protection mechanisms, depending on their type and functionality. Therefore, the different types of data that can be in a network function are modelled as subclasses of `sec:ProtectedData`. Currently there are two subclasses `sec:Software` and `sec:Secret`. Where `sec:Software` can describe pieces of code that needs protection or ML models, where `sec:Secret` can be used to create instances related to passwords, keys, tokens, etc..

```
ex:Secrets01
  a sec:Secret;
  sec:hasSecurityRequirement sec:Confidentiality
.
```

Another example, of data to be protected is an ML model, where the ML model is considered to be of a data type `sec:Software`, which is the second subclass that defines the type of data. In the below example is of an instance of a software data category which is a traffic steering ML model that has a confidentiality security requirement.

```
ex:TS_MLModel01
  a sec:Software;
  sec:hasSecurityRequirement sec:Confidentiality
.
```

5.2. Protected Identifier

The Identifier are an important asset to be protected in order to achieve privacy and ensure that for example a subscriber is not tracked or other sensitive information can be leaked due to the availability of an identifier. Thus, it's a crucial asset that has privacy requirements where cryptographic Control can be leveraged to fulfill that requirement. The `sec:ProtectedIdentifier` is a subclass of `sec:ProtectedAsset`.

Where the class sec:ProtectedIdentifier has two subclasses of the two main identifier categories that can be protected and have security requirement, `sec:SubscriberIdentifier` and `sec:MachineIdentifier` compose the main two identifier categories can be found in a network.

In the example shown below, the subscriber Identifier like SUPI, IMSI can be defined as instances of `sec:SubscriberIdentifier` and has a security requirement `sec:Privacy.` `sec:identifierType` property to be `sec:SubscriberIdentifier`

```
ex:IdentifierUPI
  a sec:SubscriberIdentifier;
  sec:hasSecurityRequirement sec:Privacy
.
```

5.3. Protected Network Interface

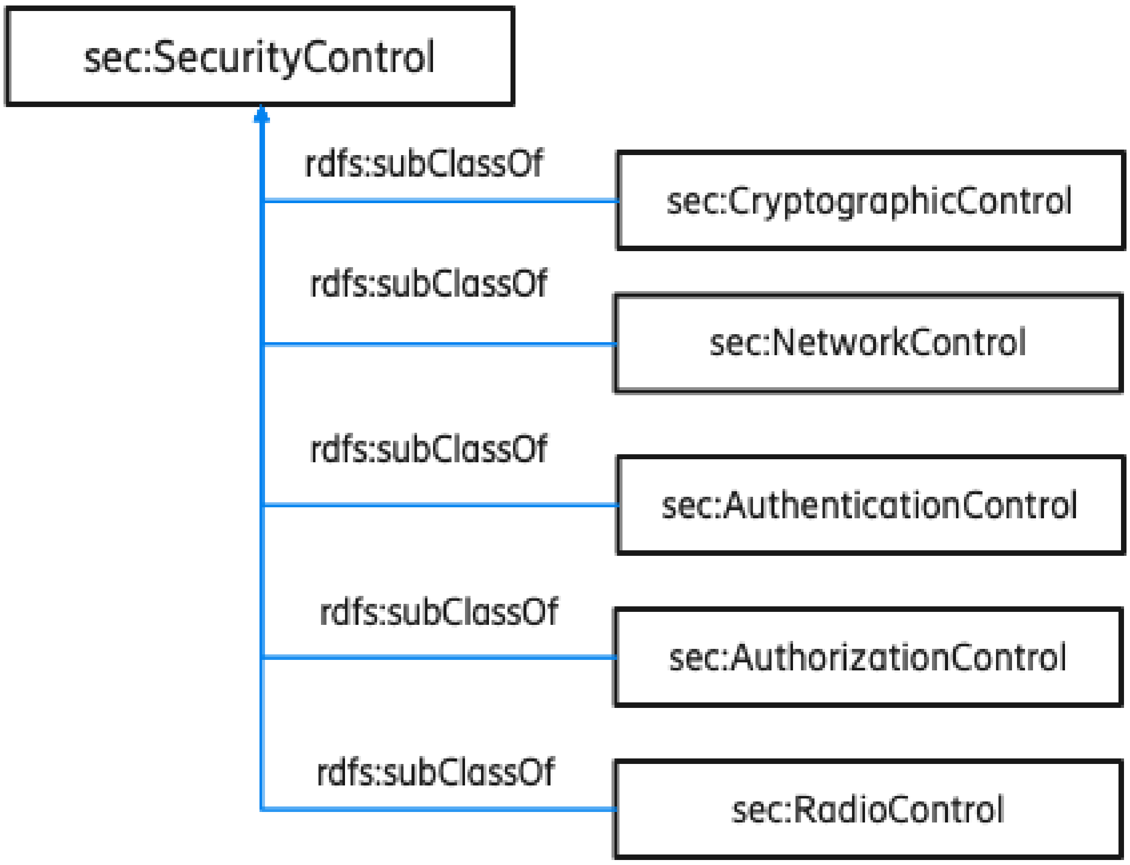
The networking stack is also considered an important asset to be protected as this is where the communications traverse between two endpoints. The network Interface in TM Forum intent ontology is defined by an instance of the class `sec:ProtectedInterface.` Since different Interface exist in a network function, each interface can have its own security requirements, hence the interface types can be part of the interface instance that is created, the property `sec:interfaceType` can be used to link to an instance of an interface in another ontology.

```
ex:UPBackhaulIf01
  a sec:ProtectedInterface;
  sec:interfaceType ex:BackhaulUP;
  sec:hasSecurityRequirement sec:Isolation
.
```

The example shown below is a resource that is defined as an instance of `sec:ProtectedInterface` that has a type of the user plane interface on the backhaul between RAN and core and has a confidentiality security requirement. The example provides a definition of the interface type using `sec:interfaceType` , this property is beneficial because it will be then inferred where isolation will be applied, and based on that the suitable security control will be deduced.

6. Security Control

In this model the security Control are presented by the class `sec:SecurityControl` . The type of security Control and the security properties they offer may vary, hence there are five subclasses of `sec:SecurityControl` that represents the main categories of security Control that can be encountered in a network (e.g., mobile network) that can be used to prevent from possible attacks. The security Control categories are represented in the subclasses `sec: CryptographicControl` , `sec:RadioControl` , `sec:NetworkControl` , `sec:AuthenticationControl` and `sec:AuthorizationControl`.

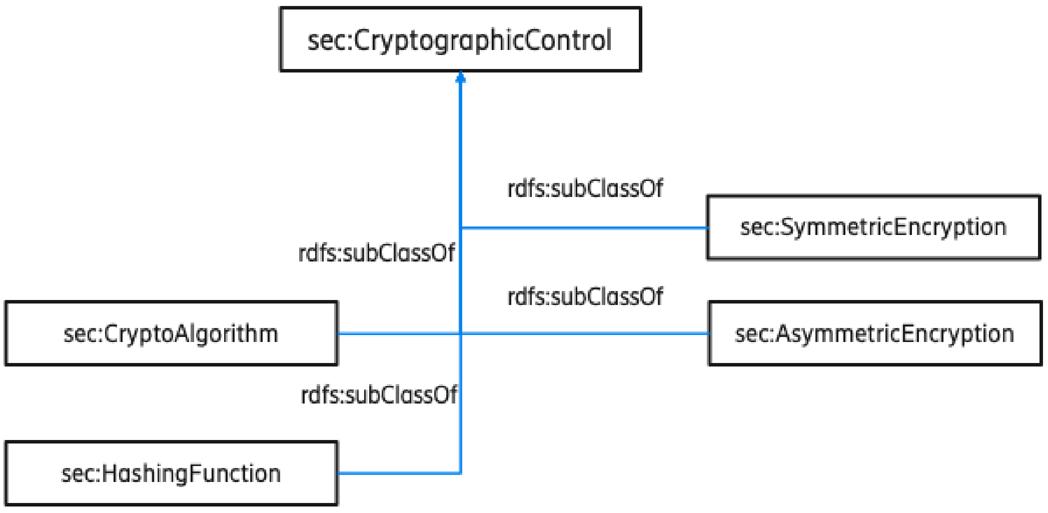


Inferring the suitable security Control can be according to the capabilities offered by the security Control and security requirements of the network or of specific Asset. The benefit of this approach is that it simplifies the way to scale the security Control. The example below illustrates that, it is expected to deliver a security Control to a set of resources that are of type `sec:ProtectedAsset`, where each protected asset can have one or more security requirement that needs to be fulfilled. Moreover, the security Control will be provisioned to those protected Asset that are in a specific geographic location. Additionally, it is required to maintain a level of security expressed by the property ex:`protectionLevel` not lower than 55%. What can be inferred is to deliver the Control that are relevant to the asset and supports the security requirement, and to continuously ensure that is effectively configured to achieve the required protection level.

```
ex:RadioControlPlane
  a sec:ProtectedInterface;
  sec:hasSecurityRequirement sec:Confidentiality,
                             sec:Integrity ;
  ex:hasStack ex:ControlPlane;
  sec:interfaceType ex:RadioInterface
.

ex:ManagementInterface
  a sec:ProtectedInterface;
  sec:hasSecurityRequirement sec:Authentication,
                             sec:Authorization;
  sec:interfaceType ex:OAM
.

ex:BackhaulInterface
  a sec:ProtectedInterface;
  sec:hasSecurityRequirement sec:Authentication,
                             sec:Confidentiality;
  ex:hasStack ex:ControlPlane,
              ex:UserPlane ;
  sec:interfaceType ex:IPInterface
.

ex:SecureNetworkgNB01
  a ex:gNB;
  sec:hasProtectedAsset  ex:RadioControlPlane,
                         ex:BackhaulInterface,
                         ex:ManagementInterface  ;
  sec:hasSecurityControl ex:SecurityControl01
.


ex:SecurityControl01
  a sec:SecurityControl;
  sec:protects [ set:resourcesOfType ( sec:ProtectedAsset );
                set:resourcesWithPropertyObject ([ sec:hasSecurityRequirement
sec:Confidentiality,
                                                                        sec:Integrity,
sec:Authentication,
sec:Authorization )]]
.

ex:T1
   a icm:Target;
   rdfs:member SecureNetworkgNB01
.

ex:DE1
  a icm:DeliveryExpectation;
  icm:target ( ex:T1 )
.

ex:protectionLevel
  rdfs:subPropertyOf icm:observable;
  tio:associatedValueType xsd:decimal
.

ex:controlCoverage
  rdfs:subPropertyOf icm:observable;
  tio:associatedValueType xsd:boolean
.

ex:PE2
  a icm:PropertyExpectation;
  icm:target ex:T1;
  log:allOf ( ex:ProtectionLevel ex:ControlCoverage)
.

ex:ProtectionLevel
   a icm:Condition;
   quan:atLeast ( [ met:lastValue ( ex:protectionLevel ) ]
                  [ rdf:value 0.55 ]
                )
.

ex:ControlCoverage
  a icm:Condition;
  quan:exactly ( [ met:lastValue ( ex:controlCoverage ) ]
                 [ true^^xsd:boolean;                quan:Quantity ]
               )
.
```

6.1. Cryptographic Security Control

Cryptographic Control are important Control that satisfy the confidentiality and integrity security requirements of an asset that requires protection. Encryption can occur in different places and different ciphers and algorithms can be used. the `sec:CryptographicControl` has several subclasses, `sec:CryptoAlgorithm` which define the type of algorithms that are used to achieve the control for example SNOW 3G encryption algorithms that is used on the radio interface or AES256 that can be used with different Control for example IPsec. The `sec:CryptoGraphicControl` has multiple subclasses that are defined as `sec:SymmetricEncryption` and `sec:AsymmetricEncryption.` For example, an instance of `sec:AsymmetricEncryption` defines a public-private encryption security control that can be leveraged for the purpose of authentication for example, or to encrypt data on an IP interface.



The security Control have different capabilities that achieve a security property or requirement , along with what type of Asset they can protect. An example, can be shown below, is the simplest for creating an object of `sec:CryptographicControl` where the defined control has a confidentiality property only and will protect the control plane stack of the radio interface. Thus, it can be inferred which protocols that this encryption algorithm shall be applied to, as well which set of encryption algorithms that supports confidentiality will be used.

```
ex:SecRadioControl01
  a sec:CryptographicControl;
  sec:protects ex:RadioControlPlane;
  sec:achieves sec:Confidentiality;
.
```

Some of the encryption Control have further profiles where each profile can either provide a different level of security or they have different computational impact on the underlying resources where they are being processed. For example, in 5G NR there are different encryption and integrity algorithms that are used as a preference list, where it is negotiated between the UE and the gNB which algorithm will be used based on the list and the UE capabilities. The radio encryption control is defined as a `rdf:Seq` since that represents an ordered list of the algorithms that will be composed of the preferred algorithms to be used in negotiations. An example can be shown below :

```
ex:encryptedThroughput
  rdfs:subPropertyOf met:metric;
  tio:associatedValueType quan:quantity
.

ex:NEA2
  a sec:CryptoAlgorithm;
  sec:algorithmKeyLength "128"^^xsd:integer;
  sec:protects ex:RadioInterface;
  sec:achieves sec:Confidentiality;
  sec:cryptoAlgorithm ex:AES128Ctr;
  ex:encryptedThroughput "58 gibi/s"^^quan:quantity
.
```

```
ex:NEA1
  a sec:CryptoAlgorithm;
  sec:algorithmKeyLength "128"^^xsd:integer;
  sec:protects ex:RadioInterface;
  sec:achieves sec:Confidentiality;
  sec:cryptoAlgorithm ex:Snow3gXor;
  ex:encryptedThroughput "48 gibi/s"^^quan:quantity
.

ex:RadioEncryption01
  a sec:CryptographiControl;
  rdf:type rdfs:Seq;
  rdf:_1   ex:NEA1;
  rdf:_2   ex:NEA2
.
```

The above example presents two different encryption algorithms defined as `ex:NEA1` and `ex:NEA2`, they are all instances of `sec:CryptoAlgorithm`. Each instance has its own properties that differentiate it from the other, for example `ex:NEA1` is defined that it is using SNOW3G as its encryption algorithm, that is presented using the property `sec:cryptoAlgorithm`, additional properties like the speed of encryption and decryption is also defined using the property `ex:encryptedThroughput` which has a `quan:quantity` data type since it's using the units "gibi/s" to present the speed of encryption in the Gigabit per second unit. `ex:NEA2` uses the same properties, and it can be shown from the example that it is using another crypto algorithm which is AES. Where both algorithms `ex:NEA1` and `ex:NEA2` are able to achieve the confidentiality security requirement.

The defined security control can be then used as part of the delivery expectation and can have its own property expectation as well, an example is provided below:

```
ex:RadioConfidentialityIntent
  a icm:Intent;
  logallOf ( ex:DE1 ex:PE2 );
  rdfs:comment "A security requirement to ensure confidentiality and integrity
protection of the radio interface"
.

ex:SecureRadioInterface
  a sec:ProtectedInterface;
  sec:hasSecurityRequirement  sec:Confidentiality,
sec:Integrity;
  sec:hasSecurityControl ex:RadioEncryption01;
  sec:interfaceType ex:RadioInterface
.

ex:gNB01
  a ex:gNB;
  sec:hasProtectedAsset  ex:SecureRadioInterface
.
```

```
ex:T1
  a icm:Target;
  rdfs:member ex:gNB01;
.

ex:T2
   a icm:Target;
   rdfs:member ex:SecureRadioInterface
.

ex:DE1
  a icm:DeliveryExpectation;
  icm:target ( ex:T1 ex:T2 )
.

ex:encryptionSpeed
  rdfs:subPropertyOf met:metric;
  tio:associatedValueType quan:quantity
.

ex:serviceLatency
  a rdf:Property;
  rdfs:subPropertyOf met:metric;
  tio:associatedValueType quan:quantity
.

ex:durationOfSession
  a rdf:Property;
  rdfs:subPropertyOf met:metric;
  tio:associatedValueType xsd:duration
.

ex:numberOfSessions
  a rdf:Property;
  rdfs:subPropertyOf met:metric;
  tio:associatedValueType xsd:integer
.

ex:PE2
  a icm:PropertyExpectation;
  icm:target ex:T2;
  log:allOf ( ex:EncryptThroughput ex:Latency ex:KeyRefresh );

.

ex:EncryptThroughput
  a icm:Condition;
  quan:atLeast ( [ met:lastValue ( ex:encryptionSpeed ) ]
                rdf:value "50 Gbps"^^quan:quantity
            )
```

```
.

ex:Latency
  a icm:Condition;
  quan:smaller ( [ met:lastValue ( ex:serviceLatency ) ]
                [ rdf:value "10"^^xsd:decimal;
                  quan:unit "ms" ]
.

ex:KeyRefresh
  a icm:Condition;
  log:anyOf ( [ quan:atMost ( ex:numberOfSessions
               [ rdf:value "10"^^xsd:decimal ]
               ) ]

             [ quan:atMost ( ex:durationOfSession
                          [ rdf:value "2"^^xsd:duration;
                            quan:unit "h" ]
                        ) ]
          )
.
```

In the previous example, the instance `ex:SecureRadioInterface` is defined as a `sec:ProtectedInterface` which had a confidentiality security requirement. Additionally, the `sec:hasSecurityControl` property was used to define a relationship between the protected asset and the instance of security control. An instance of `ex:gNb` is created which acts as one of our target to be delivered, the `ex:gNB` has the property `sec:hasProtectedAsset` which presents that `ex:SecureRadioInterface` is a part of the gNB that requires protection. This intent object expects to deliver a secure radio interface with the suitable capabilities, this is define as another target part of the `icm:DeliveryExpectation.` Additionally, in the example it was illustrated that the `sec:CryptograhicControl` have further properties to describe the encryption algorithm from an impact and performance perspective. For example, the defined `icm:PropertyExpectation` has three conditions that are related to encryption performance properties the `ex:EncrypThroughput` condition uses `quan:atLeast` to ensure that the metric `ex:encryptionSpeed` is equal or bigger than 50 Gigabit per second, this condition leads to inferring the suitable algorithm that performs at those speeds so it doesnt affect latency of the service. Moreover, a condition is defined for latency using the condition `ex:Latency` which ensures that the latency metric `ex:serviceLatency` that might be due to the impact of encryption is kept below 10 ms. Lastly, `ex:KeyRefresh` is the final condition that ensures that the key should be refreshed either if two days have passed or if the number of packet data unit (PDU) subscriber sessions has reached 10 sessions.

IP encryption Control have further properties that are used to describe the Control. IP encryption control tend to have authentication mechanisms, and they tend to be point to point, meaning that there should be two enties involved for the authentication process. An example can be illustrated below:

```
ex:IPProtectionControl01
  a sec:CryptographicControl;
  sec:protects  ex:BackhaulControlPlane01;
  sec:authenticatesWith ex:SecGW01;
  sec:cryptoControlType sec:AsymmetricEncryption
  sec:achieves sec:Confidentiality,
             sec:Integrity;
.
```

In this example, an instance of a security control `ex:IPProtectionControl01` is defined to be a `sec:CryptographicControl` which uses the property `sec:authenticatesWith` to indicate which end-point the control shall be connected to for example in the case of IPsec, and for automation this information is necessary to infer the right nodes that shall have the security control setup between them. Additionally the property `sec:cryptoControlType` is used to define which category of encryption to be used, in below example it is defined to use `sec:AsymmetricEncryption` which means that a public-private key encryption method will be used with this security control.

6.2. Network Security Control

Network Control are crucial for any network as they provide segmentation and isolation between the network functions and services to ensure more controlled access control and to reduce the lateral movement impact (east-west direction) in the network as a result of a compromise in one parts of the network. The network Control are concerened with networking stack of a system and provide protection from layer 3 up to layer 7 in the TCP/IP OSI model.

In the TMF TIO Security ontology model, the network Control are presented by `sec:NetworkControl.` Network Control provide access control to limit traffic from a source or destination, or allow bi-driectional commuincation. Thus, a network control can be further described with the following property `sec:allowTraffic`. A network control rule is composed of the tuple `<SrcHost, SrcPort, DstHost, DstPort>` to allow or deny a communication, hence `sec:allowTraffic` is modelled as a sub-property of `fun:function`, where it takes as argument types any `rdfs:Resource` that is modeled with the information of the source and destination hosts required for communication. The function has a max and min arity of 2 arguments, which represent the source (first argument) and destination (second argument) communication and it returns a result type of `rdfs:Container`, which represents the communication. An example of defining a network control can be seen below:

```
ex:NetworkSegmentationAssuranceIntent
  a icm:Intent;
  logallOf ( ex:DE3 ex:PE3 )
  rdfs:comment "An intent to ensure the right level of segmentation between
NFs"
.

ex:SecureCUCP01
  a ex:CUCP;
  sec:hasSecurityRequirement sec:Isolation;
  sec:hasSecurityControl ex:NetworkControl;
  sec:hasProtectedAsset  ex:BackhaulControlPlane01,
ex:HandOverControlPlan02
.
```

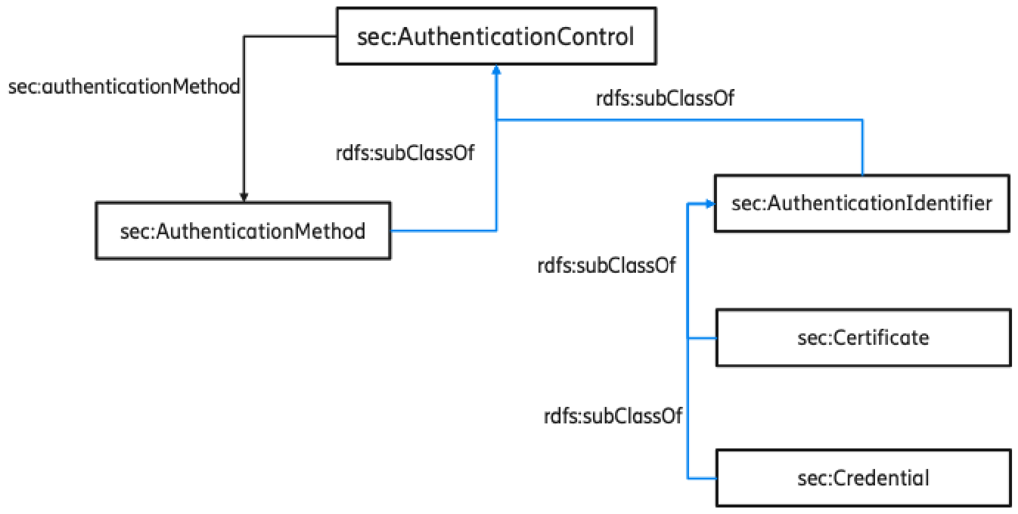
```
ex:NetworkControl
  a sec:NetworkControl;
  sec:protects  ex:BackhaulControlPlane01,
ex:HandOverControlPlan02 ;
  sec:achieves sec:Isolation;
  sec:allowTraffic ( ex:5gCN0
                     set:intersection ([set:resourcesofType (ex:gNB) ]
[set:resourcesWithPropertyObject (ex:hasLocationArea "TA#1337")]
[set:resourcesWithPropertyObject (ex:hasInterface ex:N2 ex:N3) ] )
                    )
.

ex:T1
  a icm:Target;
  rdfs:member ex:SecureCUCP01
.
ex:DE3
  icm:DeliveryExpectation;
  icm:target ( ex:T1 )
.

ex:protectionLevel01
  rdfs:subPropertyOf met:metric;
  tio:associatedValueType xsd:decimal
.
ex:ControlProtectionLevel
  a icm:Condition;
  quan:atLeast ( [ met:lastValue ex:protectionLevel01 "0.7"^^xsd:decimal ])
.

ex:PE3
  a icm:PropertyExpectation;
  icm:target ex:T1;
  log:allOf ( ex:ControlProtectionLevel )
.
```

The above example illustrates an object of type ex:SecureCUCP01 which represent a central unit control plane function which has a requirement to be delievered securely, where the sub-components or Asset in scope of protection are defined using the property `sec:hasProtectedAsset`, also the relationship is defined between the security control and the CUCP instance where the network control will be delivered and used. The instance of `sec:NetworkControl` is defined that uses the property `sec:protects` to describe which Asset shall be protected. Additionally to help inferring the right security control, the `sec:achieves` is used to define the required capabilitiy of the network control. Finally, `sec:allowTraffic` is a property that is defined as a `rdfs:subPropertyOf fun:function` , it defines the type of network functions that shall be only communicating to the Asset in scope of protection, where its arguments are the source and destination resources that should be allowed the communication. The intent has a delivery expectation to deliver a CUCP with secure network interfaces.

Moreover, the `ex:SecureCUCP01` has a condition that is defined as part of the `icm:PropertyExpectation`, the condition is the `ex:ControlProtection` which is expected to provide at least a protection level of 70% which is using the metric `ex:protectionLevel01`, that means that the network security control should be correctly configured to achieve the required protection level.

6.3. Authentication Security Control

With the increase in interconnectivity between different systems a method to correctly verify the identity of a user, process or device that is part of a communication channel is essential. In this model authentication Control are represented by the class `sec:AuthenticationControl.` Authentication focuses on the identification of a resource, the Identifier used by a resource could be credentials in the form of username or passwords or certificates. Thus, the authentication Identifier are presented in the `sec:Certificate` and `sec:Credentials` classes.



Additionally, there are two authentication methods either certificate-based or password-based where each method uses its relevant authentication identifier, thus modelling the identifier will refer eventually to which authenitcation method can be used. Depending on the method for authentication, it can provide confidentiality property as well, thus the properties that were described in the scope of `sec:CryptographicControl` can be reused with instances of `sec:AuthenticationControl` .

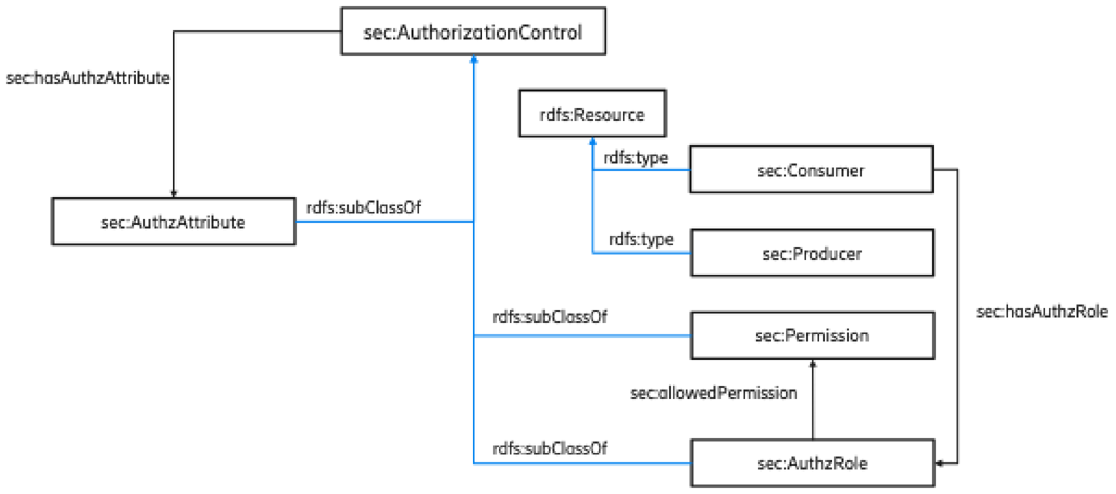
```
ex:AuthIntent
  a icm:Intent;
  log:allOf (ex:DE3 ex:PE3)
  rdfs:comment "A security requirement to authenticate SBI interfaces in
Cluster A with strong key length and a daily refresh rate of authentication
                identifier"
.


ex:CertIdentifier
  a sec:Certificate;
  sec:algorithmKeyLength "4096"^^xsd:integer;
  sec:cryptoAlgorithm ex:RSA
```

```
.

ex:AuthenticationMethod01
  a sec:AuthenticationMethod;
  sec:usesIdentification ex:CertIdentifier
.

ex:AuthenticationControl01
  a sec:AuthenticationControl;
  sec:protects ex:SBIInterface;
  sec:achieves sec:Authentication,
               sec:Confidentiality;
  sec:authenticatesWith [ set:intersection ( [ set:resourcesofType (ex:CoreNF)
];
                                             [ set:resourcesWithPropertyObject (
ex:hasLocation ex:ClusterA ) ] ) ];
  sec:authenticationMethod ex:AuthenticationMethod01;


.

ex:T2
  a icm:Target;
  rdfs:member ex:AuthenticationControl01
.

ex:DE3
  icm:DeliveryExpectation;
  icm:target ex:T2;
.

ex:PE3
  a icm:PropertyExpectation;
  icm:target ex:T2;
  sec:identifierRefresh "P1D"^^xsd:duration
.
```

In this example `ex:AuthenticationControl01` is an instance of `sec:AuthenticationControl` which is a control that achieves authenticaiton and it the same time confidentiality thus it will use encryption to secure the communication links between the network functions that have service-based Interface that is defined with `ex:SBIInterface`. It is also specified that authentication will occur only with those network functions that are of type `ex:CoreNF` and implemented in `ex:ClusterA` only. The authentication control uses sec:authenticationMethod which presents `ex:AuthenticationMethod01` as the method will be used. It can be inferred from the instance `ex:AuthenticaionMethod01` that this control will use certificate-based authentication since the type of identifier is presented by the property `sec:usesIdentification` which is an instance of `ex:CertIdentifier`. `ex:CertIdentifier` is an instance of type `sec:Certificate` and since certificates will be used to initiate the encryption handshaking and secure the communication on the SBI interface, there are some properties related to encryption defined, for example `sec:algoKeyLength` presents the length of the key that will be used by the encryption algorithm `ex:RSA`. Besides that the security control is a functional requirement, but it has a property expectation that needs to be fulfilled which presented in the property `sec:identifierRefresh`, which requires that the certificate should be renewed daily.

6.4. Authorization Security Control

Authorization is another form of access Control but rather applied on sessions that are accessing resources. In the current shift to microservices and machine to machine communication, and the adoption of zero trust architecture principles, the definition of clients and resources in the realm of authorization is extended to not only include human users but also computer resources that use any form of communication to access a service. Thus, it is crucial to ensure that services are only accessed by authorized clients and are permitted to only execute a limited set of actions on the target resource (e.g., read, write, create, execute). Authorization can be based of different parameters, that could be for example depending on the clients roles or attributes (e.g., identifier, location, etc..). Authorization it is not only limited to internal services within the network that require communication, but it can also be applied on subscribers requesting special services, for example within the same slice of a service there could be different authorization levels depending on the user categories, their subscription etc. Where based on some attributes the subscribers can access differentiated resources, specific sub-services, etc. so applying more granular authorization within a network slice.



In this model a authorization security control policy is represented using the class `sec:AuthorizationControl.` There are different examples of authorization controls where each has different purpose and its own parameters. This is reflected on the properties defined for this class. In general an authorization policy or control is concerned with the client or group of clients (i.e service consumers) that are allowed to access a target resource (i.e the service producer), the level of allowed permission for each group (i.e allowed permissions), and the attributes needed to be available in the clients in order for the authorization decision point to take a decision whether to grant authorization or deny. The main levels of permissions that are considered in this model are following the Create, Read, Update and Delete (CRUD) operation. An example is shown below:

```
ex:5GCoreAuthorizationIntent
  a icm:Intent;
  log:allOf ( ex:DE4 )
  rdfs:comment "A security requirement to ensure authorization of only
registeration and discovery service requests from AMF to NRF in Cluster A"
.
ex:5gNrf01
   a sec:Producer;
   set:intersection ( [ set:resourcesOfType ( ex:Nrf ) ]
                      [set:resourcesWithPropertyObject ( ex:hasLocation ex:ClusterA
) ]
                    );
   sec:hasSecurityRequirement sec:Authorization
.

ex:AmfRole
   a sec:AuthorizationRole;
   sec:consumerIdentity ( set:resourcesOfType ( ex:Amf ) )
   rdfs:label "AMF Role"
.

ex:Authorized5gCoreNf01
   a sec:Consumer;
   rdfs:type ex:5gCoreNf;
   sec:hasAuthorizationProfile ex:ex:AuthorizationProfile01;
   rdfs:label "5G Core NF"
 .

ex:AmfLocAttribute01
   a sec:AuthorizationAttribute;
   set:intersection ( [ set:resourcesOfType ( ex:Amf ) ]
                      [set:resourcesWithPropertyObject ( ex:hasLocation ex:ClusterA
) ]
                    ) ;
.
                                  ex:RegisterNfPerm a sec:WritePermission;
 sec:allowedPermValues ( ex:ServiceRegisterationMethods )
.

ex:QueryNfPerm a sec:ReadPermission;
 sec:allowedPermValues ( ex:ServiceDiscoveryMethods )
.

ex:AuthzCondition01
  a log:Condition;
  log:allOf ( log:match ( ex:5gCoreNf01 sec:hasAuthorizationRole ex:AmfRole)
              log:match ( ex:Amf01 sec:hasAuthorizationAttribute (
ex:AmfLocAttribute01 ) )
            )
.
```

```
ex:AuthorizationProfile01
  a sec:AuthorizationProfile;
  sec:accessTo ex:5GNRF01;
  sec:allowedPermission ( ex:RegisterNFPerm ex:QueryNFPerm
);                          log:allOf  ( ex:AuthzCondition01 );

.

ex:T1
  a icm:Target;
  rdfs:member ex:Authorized5gCoreNf01
.
ex:T2
  a icm:Target;
  rdfs:member ex:5gNrf01
.

ex:DE4
  a icm:DeliveryExpectation;
  icm:target ( ex:T1 ex:T2 )
.
```

In this example, the intent has a delivery expectation to deliver the `ex:Authorized5gNrf01` along with the approrpiate authorization controls that will fulfill the authorization requirement. A role is one of the essential concepts for authorization, where it defines the level of access of a consumer to the network and services in the network. This example focuses on a network function in the core network of role `ex:AmfRole`, it is of a type `sec:AuthorizationRole.` Usually a role is related to a user identity, thus the property `sec:consumerIdentity` describes what type of identities that will be assigned to this role, in the provided example the `ex:AmfRole` is assigned to resources with type `ex:Amf`. The AMF in this example is the consumer of the service and it is defined as a 5G core NF with `ex:5gCoreNf01` which is of a type `sec:Consumer (service requester)` and since `sec:Consumer` is modeled as a type of rdfs:Resource, the ex:5gCoreNf01 can be represented as an instance of type ex:5gCoreNf, and it is assigned the role `ex:AmfRole` using the property `sec:hasAuthzRole`. The authorization control yet can take into consideration other attributes of the consumer, it could be some specific profile infromation, location, etc.. In this example, the location of the AMF is used as its attribute to allow access to the producer. The attribute is represented by `ex:AmfLocAttribute01` which is an instance of `ex:AuthorizationAttribute`, the defined attribute takes the logical location of the AMF (i.e consumer) into asessment when making the authorization decision, only resoruces of type `ex:Amf` that are located in ex:ClusterA is allowed to require access to the NRF (i.e producer).

The authorization policy holds a information related to the resource that access is required to, the conditions that should be evaluated to grant the access and eventually what type of action will be applied once the condition evaluates to true. The property `sec:accessTo` represents to which resource this policy shall apply. Additionally, in an authorization policy it should define what type of permissions are allowed to take place on the target resource, the property `sec:allowedPermission` is used to define the set of allowed permission, `sec:allowedPermission` is a subproperty of `fun:function` which return a `rdfs:Container` that holds the set of allowed permissions. In this example only the set of permissions that are related to registeration of NF modeled as `ex:RegisterNfPerm` which has the type `sec:ReadPermission`, the other permission allowed is related to service discovery modeled as `ex:QueryNfPerm` which has the type `sec:WritePermissions`. The API based service communication can go beyond the known CRUD operations and may hold different values than the HTTP methods known for those operations, thus the allowed values that are part of those modeled permissions can be assigned to the permission using the property `sec:allowedPermValue`, it shows that the defined allowed values for each permission are the ones related to `ex:ServiceRegisterMethods` and `ex:ServiceDiscoveryMethods. ex:AuthzCondition01` is an instance of log:Condition which holds the main conditions or requireements for the authorizatin policy to grant access. It uses the logical operatior log:allOf to evaluate to "true" if all of the log:match statements return "true". The condition evaluates the statement related to the location or the attribute used along with the role assigned to the network function, if the condition evaluates to true then the access is granted which is defined with the property sec:applyAction, where the boolean value "true" means access granted, otherwise if it is "false" the access is prohibited.

7. Administrative Appendix

7.1. Document History

7.1.1. Version History

Version Number	Date Modified	Modified by:	Description of changes
3.5.0	03-May-2024	Alan Pope	Final edits prior to publication
3.6.0	04-Jul-2024	Alan Pope	Final edits prior to publication
3.7.0	14-Mar-2025	Alan Pope	Final edits prior to publication

7.1.2. Release History

Release Status	Date Modified	Modified by:	Description of changes
Pre-production	03-May-2024	Alan Pope	Initial Release
Pre-production	10-Jun-2024	Adrienne Walcott	Updated to Member Evaluated status
Pre-production	04-Jul-2024	Alan Pope	Updated to v3.6.0
Production	30-Aug-2024	Adrienne Walcott	Updated to reflect TM Forum Approved status
Pre-production	14-Mar-2025	Alan Pope	Updated to v3.7.0
Production	09-May-2025	Adrienne Walcott	Updated to reflect TM Forum Approved status

7.2. Acknowledgments

Team Member (@mention)	Company	Role*
Loay Abdelrazek	Ericsson	Author
Jörg Niemöller	Ericsson	Project Co-Chair
Kevin McDonnell	Huawei	Project Co-Chair
Yuval Stein	Amdocs	Project Co-Chair
Kamal Maghsoudlou	Ericsson	Key Contributor
Leonid Mokrushin	Ericsson	Key Contributor
Marin Orlić	Ercisson	Key Contributor
Aaron Boasman-Patel	TM Forum	Additional Input
Alan Pope	TM Forum	Additional Input
Dave Milham	TM Forum	Additional Input
Xiao Hongmei	Inspur	Reviewer

**Select from: Project Chair, Project Co-Chair, Author, Editor, Key Contributor, Additional Input, Reviewer*

8. Appendix A: Vocabulary Reference

This Appendix chapter contains a reference definition of all model vocabulary.

8.1. SecurityRequirement

The Class sec:SecurityRequirement is an abstract class of security requirements

Instance of: rdfs:Class

8.2. Authentication

The class `sec:Authentication` is abstract class of the authentication requirements

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityRequirement`

Authorization

The class `sec:Authorization` is abstract class of the authorization requirements

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityRequirement`

8.3. Availability

The class `sec:Availability` is abstract class of the availability requirements

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityRequirement`

8.4. Confidentiality

The class `sec:Confidentiality` is abstract class of the confidentiality requirements

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityRequirement`

8.5. Isolation

The class `sec:Isolation` is abstract class of the isolation requirements

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityRequirement`

8.6. Privacy

The class `sec:Privacy` is abstract class of the privacy requirements

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityRequirement`

achievedBy

The property sec:achievedBy allows defining which security control can be used to achieve a security requirement
Instance of: `rdf:Property`
Domain: `sec:SecurityRequirement`
Property: `sec:SecurityControl`

8.7. ProtectedAsset

The class sec: `ProtectedAsset` represents a resource in the network or a sub-resource in a network function that has a security requirement and requires protection

Instance of: `rdfs:Class`

8.8. hasProtectedAsset

The property sec:`hasProtectedAsset` represents the inner functions or components of a system that has security requirement.

Instance of: `rdf:Property`

Domain: `rdfs:Resource`

Range: `sec:SecurityControl`

8.9. hasSecurityControl

The property sec:`hasSecurityControl` defines the control that a resource is known to have implemented.

Instance of: `rdf:Property`

Domain: `rdfs:Resource`

Range: `sec:SecurityControl`

8.10. hasSecurityRequirement

The property sec:`hasSecurityRequirement` represents the security requirement that an asset has.

Instance of: `rdf:Property`

Domain: `sec:ProtectedAsset`

Range: `sec:SecurityRequirement`

8.11. SecurityControl

The class sec:SecurityControl represents an abstraction of a security capability or control.

Instance of: `rdfs:Class`

8.12. achieves

The property sec:`achieves` represents the type of security requirement that a control is capable of fulfilling .

Instance of: `rdf:Property`Domain: `sec:SecurityControl`
Range: `sec:SecurityRequirement`

8.13. protects

The property sec:`protects` represents the type of asset a possible control can protect.

Instance of: `rdf:Property`Domain: `sec:SecurityControl`
Range: `sec:ProtectedAsset`

8.14. CryptographicControl

The class `sec:CryptographicControl` is abstract class of the controls that apply encryption as a preventive control

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityControl`

8.15. CryptoAlgorithm

The class `sec:CryptoAlgorithm` is abstract class of the algorithms used in a crypto control

Instance of: `rdfs:Class`
Subclass of: `sec:CryptoAlgorithm`

8.16. SymmetricEncryption

The class `sec:SymmetricEncryption` is abstract class of the symmetric encryption methods

Instance of: `rdfs:Class`
Subclass of: `sec:CryptographicControl`

8.17. AsymmetricEncryption

The class `sec:AsymmetricEncryption` is abstract class of the Asymmetric encryption methods

Instance of: `rdfs:Class`
Subclass of: `sec:CryptographicControl`

8.18. HashingFunction

The class `sec:HashingFunction` is abstract class of the hashing function used as part of the cryptographic control

Instance of: `rdfs:Class`
Subclass of: `sec:CryptographicControl`

8.19. cryptoAlgorithm

The property sec`:cryptoAlgorithm` specifies a cryptographic algorithm used with a cryptographic control.

Instance of: `rdf:Property`Domain: `sec:CrypographicControl`
Range: sec:`CryptographicAlgorithm`

algorithmProperty

The property sec`:algorithmProperty` defines a capability characteristic that the cryptographic algorithm will provide, whether it is a Confidentiality or Integrity to satisfy the security requirements.

Instance of: `rdf:Property`Domain: `sec:CryptoAlgorithm`
Range: `sec:SecurityRequirement`

8.20. algorithmKeyLength

The property sec`:algorithmKeyLength` defines the key length characteristic of the cryptographic algorithm.

Instance of: `rdf:Property`Domain: `sec:CryptoAlgorithm`
Range: `xsd:integer`

8.21. NetworkControl

The class `sec:NetworkControl` is abstract class of the controls that apply protection on the network level

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityControl`

8.22. allowTraffic

The property `sec:allowTraffic` is a sub-property of func:function specifies the assets a protected asset can communicate with in the outbound direction. The arguments provided to the function represent the peers of the communication, where source is in the first position, the destination is in the second position and third position is optional to reflect the action whether to permit or allow. It returns a result type of rdfs:Container that holds the ifnromation on the source host, destination host and orientation of the traffic whether it is bi-directional or uni-driectional.

Instance of: `rdf:Property`

Domain: `sec:NetworkControl`

SubProperty of: `func:function`

Minimum Arity: 2

Maximum Arity: 3

Result Type: `rdfs:Container`

8.23. AuthenticationControl

The class `sec:AuthenticationControl` is abstract class of the authentication control

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityControl`

8.24. AuthenticationIdentifier

The class `sec:AuthenticationIdentifier` is abstract class of the authentication identifier used for the process of authentication for example, username, certificates, subscriber identifiers, etc..

Instance of: `rdfs:Class`
Subclass of: `sec:AuthenticationControl`

8.25. authenticationMethod

The property sec:`authenticationMethod` specifies the authentication method an authentication control is using. Authentication methods can also be used by other types of security controls that have authentication capabilities embedded within.

Instance of: `rdf:Property`Domain: `sec:SecurityControl`
Range: `sec:AuthenticationControl`

8.26. authenticateWith

The property sec:`authenticateWith` specifies the corresponding party or the authentication server the authentication control should perform the authentication procedures with.

Instance of: `rdf:Property`Domain: `sec:AuthenticationControl`
Range: `sec:ProtectedAsset`

8.27. AuthorizationControl

The class `sec:AuthorizationControl` is abstract class of the authorization control

Instance of: `rdfs:Class`
Subclass of: `sec:SecurityControl`

8.28. Consumer

The class `sec:Consumer` is abstract class to represent the source of the authorization request

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.29. Producer

The class `sec:Producer` is abstract class to represent the target of the authorization request

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.30. accessTo

The property sec:`accessTo` specifies the corresponding party or the authentication server the authentication control should perform the authentication procedures with.

Instance of: `rdf:Property`Domain: `sec:Consumer`
Range: `sec:Producer`

Permission

The class `sec:Permission` is abstract class to represent the permission allowed to a authroization request source to access a request target.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.31. CreatePermission

The class `sec:CreatePermission` is abstract class to represent the create operation following the CRUD operations.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.32. ReadPermission

The class `sec:ReadPermission` is abstract class to represent the read operation following the CRUD operations.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.33. UpdatePermission

The class `sec:UpdatePermission` is abstract class to represent the update operation following the CRUD operations.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.34. DeletePermission

The class `sec:DeletePermission` is abstract class to represent the delete operation following the CRUD operations.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.35. allowedPermValue

The property sec:`allowedPermValue` specifies the specific permission values part of the permission operations.

Instance of: `rdf:Property`Domain: `sec:Permission`

8.36. AuthorizationRole

The class `sec:AuthorizationRole` is abstract class for the role of the source of an authorization request

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.37. consumerIdentity

The property sec:`consumerIdentity` specifies the identifier of the authorization request source.

Instance of: `rdf:Property`Domain: `sec:AuthorizationRole`

8.38. hasAuthorizationRole

The property sec:`consumerIdentity` specifies the identifier of the authorization request source.

Instance of: `rdf:Property`Domain: `sec:Consumer`

Range: sec:AuthorizationRole

8.39. AuthorizationAttribute

The class `sec:AuthorizationAttribute` is abstract class for extra attributes that can be used for authorization to add further context to the authorization request.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.40. hasAuthorizationAttribute

The property sec:`hasAuthorizationAttribute` specifies the attribute in the authorization request to be checked further in an authorization policy.

Instance of: `rdf:Property`Domain: `sec:Consumer`

Range: `sec:AuthorizationAttribute`

8.41. AuthorizationProfile

The class `sec:AuthorizationProfile` is abstract class for an authorization profile.

Instance of: `rdfs:Class`
Subclass of: `sec:AuthorizationControl`

8.42. allowedAuthorizationRole

The property sec:`allowedAuthorizationRole` specifies the allowed roles to access a target resource.

Instance of: `rdf:Property`Domain: `sec:AuthorizationProfile`

Range: `sec:AuthorizationRole`

8.43. allowedPermission

The property sec:`allowedPermission` specifies the allowed permissions and operations to be applied on a target resource.

Instance of: `rdf:Property`Range: `sec:Permission`

8.44. hasAuthorizationProfile

The property sec:`hasAuthorizationProfile` specifies the authorization profile associated with an instance of a service consumer .

Instance of: `rdf:Property`Domain: `sec:Consumer`

Range: `sec:AuthorizationProfile`