ETSI TR 103 534-1 V1.1.1 (2019-08)

SmartM2M;
Teaching material;
Part 1: Security

Reference
DTR/SmartM2M-103534-1

Keywords
cybersecurity, IoT, oneM2M

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

The present document can be downloaded from:
http://www.etsi.org/standards-search

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx

If you find errors in the present document, please send your comment to one of the following services:
https://portal.etsi.org/People/CommiteeSupportStaff.aspx

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.
The content of the PDF version shall not be modified without the written authorization of ETSI.
The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2019.
All rights reserved.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members.
3GPP™ and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
oneM2M™ logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners.
GSM® and the GSM logo are trademarks registered and owned by the GSM Association.

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

The present document is part 1 of a multi-part deliverable covering SmartM2M Training Material, as identified below:

Part 1: "Security";

Part 2: "Privacy".

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

1.1 Context for the present document

The design, development and deployment of - potentially large - IoT systems require to address a number of topics - such as privacy, interoperability or security - that are related and should be treated in a concerted manner. In this context, several Technical Reports have been developed that each address a specific facet of IoT systems.

In order to provide a global a coherent view of all the topics addressed, a common approach has been outlined across the the present document concerned with the objective to ensure that the requirements and specificities of the IoT systems are properly addressed and that the overall results are coherent and complementary.

The present document has been built with this common approach also applied in all of the other documents listed below:

ETSI TR 103 533 [i.1]

ETSI TR 103 534 [i.15]

NOTE: ETSI TR 103 534-1 is the present document

ETSI TR 103 535 [i.3]

ETSI TR 103 536 [i.4]

ETSI TR 103 537 [i.5]

ETSI TR 103 591 [i.19]

1.2 Scope of the present document

The present document presents teaching material to allow readers, identified by role, to gain knowledge of the fundamentals of IoT security.

The present document is structured as a set of annexes each containing the outline of training material. The more detailed training material, in the form of a set of PowerPoint slides is provided in archive tr_10353401v010101p0.zip as an electronic addition to the present document.

The annexes contain training material in the following areas:

- Threat, Vulnerability and Risk Analysis (TVRA) in IoT:
  - The role of TVRA is primarily to ensure that a system is designed and deployed with a thorough understanding of the environment in which it will be deployed, the purpose of the system, the components or assets of the system, the links between the deployment and its environment, and the technical/procedural/regulatory basis of the system. Having this core understanding alongside an analysis of the threats and threat agents that will seek to attack the system leads to an understanding of the risks to the system.
  - The material in this clause extends from material prepared for the ETSI TVRA Workshop (March 2009) and is based on the TVRA method published in ETSI TS 102 165-1 [i.2] with specific IoT use cases to drive the TVRA exercise.
- Secure configuration of IoT devices:
  - The vast majority of security failures occur as a result of poor configuration. For example reliance on default security attributes (the default password conundrum). The purpose of this module is to give guidance on how to securely configure IoT devices to minimise their attack surface.

- Cryptographic security basics as they apply in IoT:
  - Cryptography is the mathematical toolset that underpins the majority of countermeasures (i.e. authentication, encryption, integrity proof and verification). The purpose of this module is to give a simple grounding in the role and purpose, and the underlying mechanisms of cryptography. Amongst the topics to be covered are the following:
    - Role of cryptography in security
    - Historic roots of cryptography
    - Relationship identification to pre-select cryptographic architecture
    - Core cryptographic modes
  - The material provides examples based on AES as published in FIPS 197 [i.11] and the Diffie Hellman asymmetric key exchange protocol.
- Secure operation of IoT devices:
  - Closely related to secure configuration is secure operation and this module addresses the measures required to assure that a securely configured device can be operated securely.
- Applying best practices to IoT security:
  - The purpose of this module is to give specific training in how to apply the best practices identified in ETSI TR 103 533 [i.1] to real IoT systems.
- Programming guide for secure IoT:
  - The purpose of this module is to give guidance on secure or safe programming. By means of coding examples (in programming languages including Swift, C, C++, Java) the steps to minimise security flaws in programming of IoT devices.
- Guide to selecting a training provider:
  - A guide to the identification and selection of training providers and training programmes in IoT.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 103 533: "SmartM2M; Security; Standards Landscape and best practices".

[i.2] ETSI TS 102 165-1: "CYBER; Methods and protocols; Part 1: Method and pro forma for Threat, Vulnerability, Risk Analysis (TVRA)".

[i.3] ETSI TR 103 535 (V1.1.1): "SmartM2M; Guidelines for using semantic interoperability in the industry".

[i.4] ETSI TR 103 536: "SmartM2M; Strategic / technical approach on how to achieve interoperability/interworking of existing standardized IoT Platforms".

[i.5] ETSI TR 103 537: "SmartM2M; Plugtests™ preparation on Semantic Interoperability".

[i.6] ETSI TR 103 591: "SmartM2M; Privacy study report; Standards Landscape and best practices".

[i.7] AIOTI: "High Level Architecture (HLA)", Release 4.0, June 2018.

[i.8] ENISA: "IoT Security Standards Gap Analysis", ISBN: 978-92-9204-275-2, DOI: 10.2824/713380.

[i.9] Regulation (EU) 2019/881 of the European Parliament and of the Council of 17 April 2019 on ENISA (the European Union Agency for Cybersecurity) and on information and communications technology cybersecurity certification and repealing Regulation (EU) No 526/2013 (Cybersecurity Act).

[i.10] ETSI TS 103 645: "CYBER; Cyber Security for Consumer Internet of Things".

[i.11] National Institute of Standards and Technology (NIST) FIPS 197: "Federal Information Processing Standards Publication 197; Advanced Encryption Standard (AES)", November 26, 2001.

[i.12] GSMA IoT Security Guidelines and IoT Security Assessment.

NOTE: Available from https://www.gsma.com/iot/iot-security/iot-security-guidelines/

[i.13] ETSI TR 103 305-1: "CYBER; Critical Security Controls for Effective Cyber Defence; Part 1: The Critical Security Controls".

[i.14] ETSI TR 103 534: "SmartM2M; Teaching Material: Part 1 (Security) and Part 2 (Privacy)".

NOTE: ETSI TR 103 534-1 is the present document.

[i.15] ETSI EN 300 392-7: "Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 7: Security".

3 Definition of terms, symbols and abbreviations

3.1 Terms

Void.

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES	Advanced Encryption Standard
AIOTI	The Alliance for Internet of Things Innovation
API	Application Programming Interface
CASP	CompTIA Advanced Security Practitioner
CIA	Confidentiality, Integrity, Availability
CISSP	Certified Information Systems Security Professional
DB	DataBase

DBA	DataBase Administator
DBMS	DataBase Management System
DDoS	Distributed Denial of Service
DevOps	Development IT Operations
DevSecOps	Secure DevOps
DNS	Domain Name System
DoS	Denial of Service
e-CF	European e-Competence Framework
ENISA	European Union Agency for Network and Information Security
ERP	Enterprise Resource Planning
ESAPI	OWASP Enterprise Security API
ETSI	European Telecommunication Standards Institute
FIPS	Federal Information Processing Standard
GDPR	General Data Protection Regulation
GSMA	GSM Association
IaC	Infrastructure as Code
ICT	Information and Communications Technology
IoT	Internet of Things
ISC^2	International Information System Security Certification Consortium
IT	Information Technology
ORM	Object Relational Mapper
OS	Operating System
OWASP	Open Web Application Security Project
RSA	Rivest Shamir Aldeman
SQL	Structured Query Language
SSCP	Systems Security Certified Practitioner
TOE	Target Of Evaluation
TR	Technical Report
TVRA	Threat, Vulnerability and Risk Analysis

4 What is Security?

4.1 Introduction and overview

The question "What is Security?" is very difficult to answer succinctly. In the context of ICT, security is often taken to refer to the prevention of various forms of attack on the system, or elements of the system. The purpose of the present document, as indicated in the Scope statement, is to provide material to allow readers, identified by role, to gain knowledge of the fundamentals of IoT security.

Whilst these concepts are expanded upon in the remainder of the present document the complexity of "security" as a topic to understand is highlighted by the many roles and process that "security" has to tackle:

- System protection role:
  - Core CIA roles - least knowledge model to assure system operation.
  - Analytic role - data required to forecast, resolve, recover.
- Anti-adversary role:
  - Identify who gains from system breaches.
- Risk management role:
- Regulatory compliance role:
  - Assurance of technical provisions for GDPR, for the Cyber-Security directive, for law enforcement, for support of the eIDAS regulation and so forth.

A reasonable level of understanding of each of these roles, and the technologies and processes that enable them, is the ultimate goal of the present document.

4.2 Teaching goals

The bulk of the material in the present document is aimed at tutor led teaching and there is a presumption of prior knowledge to apply the material to the actual audience. Thus there are hints given at points in the material for the tutor/teacher to drive classroom exercises. Such exercises are not definitive in that there is no implied certificate or other statement of learning from the material but are intended to allow the tutor to assess the success of students in assimilating the material offered.

NOTE: In the case of tutor lead classwork the tutor is expected to expand upon the base material that is provided in the present document and its attachments as required by the students.

4.3 Learning goals

Whilst not specifically designed for self-tutoring when used in such a context, as for teaching goals, the present document has some specific learning goals when acting as the basis of self-taught material. Specific learning goals are indicated at the start of each clause in order to guide the reader as to the new knowledge that will be gained after completion of the material in each clause.

5 Security in the context of IoT

5.1 A global approach to IoT Systems

5.1.1 Major characteristics of IoT systems

IoT systems are often seen as an extension to existing systems needed because of the (potentially massive) addition of networked devices. However, this approach does not take stock of a set of essential characteristics of IoT systems that push for an alternative approach where the IoT system is at the centre of attention of those who want to make them happen. This advocates for an "IoT-centric" view.

Most of the above-mentioned essential characteristics may be found in other ICT-based systems. However, the main difference with IoT systems is that they all have to be dealt with simultaneously. The most essential ones are:

- **Stakeholders.** There is a large variety of potential stakeholders with a wide range of roles that shape the way each of them can be considered in the IoT system. Moreover, none of them can be ignored.
- **Privacy.** In the case of IoT systems that deal with critical data in critical applications (e.g. e-Health, Intelligent Transport, Food, Industrial systems), privacy becomes a make or break property.
- **Interoperability.** There are very strong interoperability requirements because of the need to provide seamless interoperability across many different systems, sub-systems, devices, etc.
- **Security.** As an essential enabling property for Trust, security is a key feature of all IoT systems and needs to be dealt with in a global manner. One key challenge is that it is involving a variety of users in a variety of use cases.
- **Technologies.** By nature, all IoT systems have to integrate potentially very diverse technologies, very often for the same purpose (with a risk of overlap). The balance between proprietary and standardised solutions has to be carefully managed, with a lot of potential implications on the choice of the supporting platforms.
- **Deployment.** A key aspect of IoT systems is that they emerge at the very same time where Cloud Computing and Edge Computing have become mainstream technologies. All IoT systems have to deal with the need to support both Cloud-based and Edge-based deployments with the associated challenges of management of data, etc.
- **Legacy.** Many IoT systems have to deal with legacy (e.g. existing connectivity, back-end ERP systems). The challenge is to deal with these requirements without compromising the "IoT centric" approach.

5.1.2 The need for an "IoT-centric" view

5.1.2.1 Introduction

In support of an "IoT-centric "approach, some elements have been used in the present document in order to:

- Support the analysis of the requirements, use cases and technology choices (in particular related to interoperability).
- Ensure that the target audience can benefit from recommendations adapted to their needs.

5.1.2.2 Roles

A drawback of many current approaches to system development is a focus on the technical solutions, which may lead to suboptimal or even ineffective systems. In the case of IoT systems, a very large variety of potential stakeholders are involved, each coming with specific - and potentially conflicting - requirements and expectations. Their elicitation requires that the precise definition of roles that can be related to in the analysis of the requirements, of the use cases, etc.

Examples of such roles to be characterised and analysed are: System Designer, System Developer, System Deployer, End-user, Device Manufacturer. Some of these roles are specifically addressed in the present document.

5.1.2.3 Reference Architecture(s)

In order to better achieve interoperability, many elements (e.g. vocabularies, definitions, models) have to be defined, agreed and shared by the IoT stakeholders. This can ensure a common understanding across them of the concepts used for the IoT system definition. They also are a preamble to standardisation. Moreover, the need to be able to deal with a great variety of IoT systems architectures, it is also necessary to adopt Reference Architectures, in particular Functional Architectures. The AIOTI High-Level Architecture (see [i.7]) is the reference for the present document.

5.1.2.4 Guidelines

The very large span of requirements, Use Cases and roles within an IoT system make it difficult to provide prototypical solutions applicable to all of the various issues addressed. The approach taken in the present document is to outline some solutions but also to provide guidelines on how they can be used depending on the target audience. Such guidelines are associated to the relevant roles and provide support for the decision-making.

6 Overview of IoT security challenge

6.1 The challenge

The core challenge in IoT, particularly embedded IoT where devices are often of an "enable and forget" form, is to recognise that as an IoT device has processing and communication capacity, that often can be programmed post-shipment, that it is a vector to attack any of the user, the system or something in the wider connected network. This entails understanding of the management of risk through management of impact (if something happens) and management of likelihood (of something happening). These aspects, impact and likelihood of an attack, have been at the core of security engineering since its inception. The set of things that engineers are able to do to minimise risk are considerable and part of the challenge in IoT is to provide a minimum set of technical capabilities that maximise the security of the system. However even with the appropriate technology in place it is still necessary to have the non-technology aspects correctly implemented and this means distribution chains, physical security measures, personnel measures and so forth.

The primary security provisioning strategies of redesign or hardening are consistent with the goal of security design to ensure a low likelihood of an unwanted incident arising. As the likelihood of an unwanted incident is dependent upon the presence of weakness in an asset and also the presence of both threats and threat agents that exploit the weakness it is the purpose of security systems to remove, or mask, the weaknesses of an asset.

- Asset redesign:
  - The assumption made prior to analysis is that all assets have weaknesses and the job of the analyst is to identify those weaknesses. Where weaknesses are found and have a large number of associated threats and threat agents there may be a possibility to redesign the asset in such a way as to remove the inherent weaknesses. The viability of this strategy will depend on a number of factors including the maturity of the asset design and the relative cost of redesign versus the cost of weakness masking through asset hardening.
- Asset hardening:
  - An asset may have some weaknesses that cannot be removed but which may be masked or made inaccessible by the addition of additional features or capabilities to the vulnerable asset or other assets in the system such that the combination of assets in the system presents a lower likelihood of attack, and hence a lower risk to the system.

EXAMPLE: A firewall is a classical asset hardening technique in that it does not alter the system behind the wall but makes the asset or system hard to get to as the attacker first has to get past the wall.

6.2 Conventions and terminology

Security documents, and the engineers and designers, often use a distinctive terminology. The primary stakeholders are given names: Alice, Bob and Eve. Alice and Bob are the parties to a secure transaction, so in general terms Alice is trying to establish a relationship with Bob. Eve is the generalised adversary, whose purpose is to attack the relationship between Alice and Bob.

One of the common models is to consider security in broad terms as determination of the triplet {threat, security-dimension, countermeasure} such that a triple such as {interception, confidentiality, encryption} is formed. The threat in this case being interception which risks the confidentiality of communication, and to which the recommended countermeasure is encryption. More detail discussion of the security dimensions is given below. However, the nature of threats needs greater examination. Threats can be gathered into several families:

- **Masquerade:** In this case Eve attempts to impersonate Bob such that Alice will interact with Eve in the belief that she is talking to Bob. Masquerade attacks are most often countered by the use of authentication schemes.
- **Manipulation:** In this case Eve attempts to modify data that Alice is trying to access (generally given to Alice by Bob). Examples include falsifying location, falsifying an account balance, introducing malicious code to an application. Countering manipulation is often difficult but includes the use of cryptographic checksums, error detection and correction codes, digital signatures. The security problem is to ensure that identification and proof of manipulation cannot be manipulated too.
- **Interception:** In this case Bob is sending something to Alice and Eve views it in transit. Conventionally interception threats are countered by encryption (in which case Eve can see that communication exists between Alice and Bob but is unable to see the content of that communication).
- **Theft:** The theft attack extends the interception attack in that whilst Eve not only observes the communication between Alice and Bob she may also intercept and redirect such that Alice never receives the data, or copies the data such that whilst Alice receives the data Eve also has the data. The means to counter theft include various forms of access control, encryption (if data is stolen it cannot be read), manipulation control including things such as proof of source (if stolen data is re-presented this uses techniques that can reveal it does not actually belong to the presenting party).

A further set of terms in security relate to the nature of an attack. Attacks can be direct, but can also be indirect. A direct attack is just that, Eve directly attacks Alice or Bob or the link between them. However, Eve can also attack Alice, Bob or their relation by attacking something else altogether, these are termed side-channel attacks. For example in the internet there are several tools and protocols used to ensure that Alice and Bob can connect, so Eve can attack one of those tools or protocols and prevent Bob from ever being able to attach to Alice, this could be achieved by poisoning the DNS system, or attacking the network in such a way that valid traffic never reaches Alice (a Denial of Service (DoS) attack, or a Distributed Denial of Service (DDoS) attack).

In addition to attacks against device function there are a class of attacks that use devices owned by Alice to act as adversaries on behalf of Eve.Commonly these are referred to as "Trojan" devices, in reference to the myth of the gift of the wooden horse from the Greeks to the people of Troy containing a hidden cadre of Greek soldiers who when inside the city launched an attack (in other words an apparently benign package contains a hidden payload used to attack the host).

6.3 Trust and roots of trust

Security mechanisms and procedures are predicated on some form of trust. There is equivalence in the real world of course: Alice trusts the bank to store her money and to make it available on demand; Alice trusts that the bank will not lose her money. The reasons that Alice trusts her bank are not always explicit but are a combination of peer review and peer experience, of knowledge that banks are overseen by a regulatory authority, of independent reviews.

In the security domain the concept of trust is complex and multi-tiered. For an analogy to examine the levels of complexity the use case or example considers that Alice is travelling across town to make a deposit of cash (very large amount) in her bank. As cash is a highly desirable and easy to dispose of item it is attractive to thieves to steal, so Alice will take steps to not-advertise that she is carry a lot of cash. She will also take steps to know exactly the form in which she is carrying the cash, e.g. the denomination of each note and the number of each note, and she will keep this record separate from the cash itself. Alice has to trust that the cash she is carrying is legitimate so she trusts that she has received them from a reputable source. If Alice has a small window of time to take the cash to the bank she has to trust her transport. In choosing transport she has to make a number of decisions regarding speed of transfer, safety of transfer, cost of transfer, and exposure during transfer to Eve. Alice may choose public transport, or private transport, she may choose to walk, to cycle, to drive. Each of these offers a different level of risk and places the cash she is transferring at a different level of risk too. In determining which transport option to select Alice has to determine which she trusts most to get her to the bank in time with lowest risk to the cash she is carrying. It would be easy to continuously extend the scenario and in a teaching context the teacher is recommended to perform a brainstorming exercise with the class to tease out the trust relationships that Alice has to maintain in the overall process of taking cash from home to safely depositing it in the care of the bank.

Teaching exercise: Get students to build a map of the trust relationships for the scenario, and to then explain the risks in terms of Eve and the residual risk accepted by Alice.

6.4 The CIA paradigm

Notwithstanding the discussion across ETSI TR 103 533 [i.1] the purpose of security technologies is multi-fold:

- **Confidentiality:** Information shared by Alice with Bob is only visible to Bob and Alice. If Eve can access the information she cannot ascertain the meaning of the content. Primarily achieved using encryption.
- **Integrity:** Information shared by Alice with Bob can be proven by Alice not to have been manipulated by a 3rd party (Eve). Bob can verify this is the case. Primarily achieved using hash functions with have specific characteristics.
- **Availability:** This addresses the aim of ensuring that an authorized party (i.e. Alice) is able to access services or information when needed. In other words, that Alice has access only to those assets she is allowed to access and that they are available to Alice when legitimately demanded, and that an adversary, Eve, does not have access. The technologies that address this include Identity Management, Authentication and Access Control, in addition considerations in the availability domain include reliability and resilience which, whilst not strictly addressed by security technology, impact on availability.

One of the many characteristics of IoT is that the number of communicating entities is very large and the number of possible relationships per device is larger than, say, cellular telecommunication where in effect the mobile device connects to a trusted network using credentials held by their home, trusted, service provider.

As a trivial example it can be imagined that IoT communications security is equivalent to sending presents to somebody. To ensure the recipient does not know the content before unwrapping, the sender masks the content by wrapping the gift - this makes the content confidential. The intended recipient is clearly indicated on the label as is the sender - this identifies the parties to the transaction and depending on how names are written may confer some proof of identity. Finally, in order to ensure the package is not damaged the sender adds packaging that protects it - this is some means of ensuring the integrity of the package is maintained in transit. Translating this to IoT data from A to B the data can be encrypted to confer confidentiality, the parties A and B have to be able to prove their identity to confer authenticity to the exchange, and the parties can add data to the package that will be used to assure and verify the integrity of the package.

There are a number of complexities in IoT that arise from the nature and number of both devices and connections. The most obvious of these is key management.

6.5 Review of landscape and best practices

A summary before consideration of using any security guidance or best practice is to reflect on the purpose of security technologies and processes.

- System protection role:
  - Core CIA roles - least knowledge model to assure system operation.
  - Analytic role - data required to forecast, resolve, recover.
- Anti-adversary role:
  - Identify who gains from system breaches.
- Risk management role.
- Regulatory compliance role:
  - Assurance of technical provisions for GDPR, for Cyber-Security directive, for law enforcement, etc.

In looking at best practice addressed to non-security professionals, e.g. to developers to ensure that they have designed the necessary capability (the power or ability to do something) and functionality (the quality of being suited to serve a purpose well) into their products and services, to managers to ensure they have recruited the necessary expertise, to users to ensure they maximise the available features of their devices, the guidance and best practices have to address some or all of the roles in the foregoing list. Guidelines and best practices should be written in such a way that they recognise the overlap of each of the roles. For example in looking at guidance that limits the attack surface exposed to an adversary (anti-adversary role) it is necessary to consider guidance on how to achieve proof of who is acting in the system (system protection and CIA roles) but as this may have an impact on regulatory compliance the guidance in that role needs to be considered, also if data is required to protect the system by analysis of how the system is used then the guidelines for system protection by analytics needs to be taken into account.

6.6 Rationale for training and education in IoT security

The ENISA gap analysis [i.8] identified a significant gap in the understanding of security and its application to systems. The present document and its included training material is one means of addressing that perceived gap. However a further strong message, made in ETSI TR 103 533 [i.1], is that whilst in many fields there is an ability to learn "on the job", for security there is no such luxury. A danger that is not expressed in any of the guidelines cited in ETSI TR 103 533 [i.1] is the consequence of incomplete implementation, or of incomplete knowledge. Similarly even if all of the teaching material given in, and cited by, the present document, is followed the resulting knowledge should never be treated as complete, and certainly not sufficient to combat an equally trained adversary.

NOTE: As a result of the publication of Regulation (E) 2019/881 [i.9] ENISA will henceforth be known as the EU Agency for Cybersecurity, It is assumed that publications from ENISA will be maintained and remain available in the new regime.

The world of security is often downplayed as non-adversarial. This is not the approach taken in the present document. Rather it is assumed that the system is never benign and that adversaries are always at work.

7 Security use cases

In IoT, as in most systems once they have been sufficiently decomposed, the core idea is that an entity, referred to as Alice, wants to exchange information with another entity, referred to as Bob, in the presence of an adversary, Eve. In the world of IoT certain assumptions can be made regarding Alice and Bob, and about the means by which they are connected.

The thought experiment that will underpin all of the material in the present document is the following:

"*How can Alice transfer something to Bob in such a way that she is assured that only Bob will receive it, that Eve is unable to masquerade as Bob, that Eve cannot see what Alice is transferring, and where Eve is unable to manipulate what Alice is transferring?*"

The architecture and mechanics of security are to some extent determined by the set of known variables in the thought experiment. This is captured, somewhat clumsily but accurately in the statement attributed to Donald Rumsfeld as "*There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know.*" In designing and understanding security the knowledge of the security designer has to be such that the known knowns are maximised, and the set of unknown knowns is minimised. This means knowing as much as possible about each of Alice and Bob and how they connect, and of how Eve could subvert the system. In recognising that there are known unknowns the designer has to be able to minimise any impact they would have, and this is essentially to assume that Eve will try new ways to attack the system. Thus whilst it is not possible to know exactly how Eve will operate, the system can be designed to limit the degrees of freedom that Eve has at her disposal.

An everyday example of the thought experiment is the sending of a gift. Alice wishes to send Bob a gift. In order to make sure it gets to Bob she addresses the package such that there is no ambiguity in who it is being sent to. She may ask for proof of delivery. She may ask for a trusted carrier to carry the package. In order to prevent Eve seeing what is in the gift Alice will wrap it, and to ensure it is not broken in transit Alice will also provide protective wrapping (bubble wrap perhaps). Each of these actions is a security action: Making sure Bob is the recipient is the act of Identification and Authentication; Concealing the gift is providing confidentiality to the gift; Bubble wrap acts to preserve the integrity of the package. This is the everyday application of the CIA paradigm that underpins almost all security work.

Annex A:
Threat, Vulnerability and Risk Analysis in IoT

A.1 Role of TVRA

The purpose of the TVRA exercise is to identify where risks exists, to classify the level of the risk, and to guide the application of countermeasures that manage risk to an acceptable residual value.

The TVRA method, or process, consists of the following steps:

1) Identification of the Target of Evaluation (TOE) resulting in a high level description of the main assets of the TOE and the TOE environment and a specification of the goal, purpose and scope of the TVRA.

2) Identification of the objectives resulting in a high level statement of the security aims and issues to be resolved.

3) Identification of the functional security requirements, derived from the objectives from step 2.

4) Inventory of the assets as refinements of the high level asset descriptions from step 1 and additional assets as a result of steps 2 and 3.

5) Identification and classification of the vulnerabilities in the system, the threats that can exploit them, and the unwanted incidents that may result.

6) Quantifying the occurrence likelihood and impact of the threats.

7) Establishment of the risks.

8) Identification of countermeasures framework (conceptual) resulting in a list of alternative security services and capabilities needed to reduce the risk.

9) Countermeasure cost-benefit analysis (including security requirements cost-benefit analysis depending on the scope and purpose of the TVRA) to identify the best fit security services and capabilities amongst alternatives from step 8.

10) Specification of detailed requirements for the security services and capabilities from step 9.

Many of the terms used in the TVRA are specialised thus need greater examination which is provided throughout this annex.

A.2 Identification of IoT Security environment

In general terms an IoT device requires connectivity to services or components across a network. This IoT pre-requisite implies the existence of at least one open interface that can be used to attack the IoT device. It is also implied that the network expects connections from IoT devices which again implies an open interface that can be used to attack the IoT dependent infrastructure.

The identification of the IoT security environment establishes the Target of Evaluation (TOE), that is the area in which each of Alice, Bob and Eve are considered to operate. The result is a high level description of the main assets of the TOE and the TOE environment and a specification of the goal, purpose and scope of the TVRA which by implication is the IoT security environment.

This addresses steps 1, 2, 3 and 4 of the method/process.

A.3 Modelling of threats and vulnerabilities

This addresses steps 5 and 6 of the method/process.

A vulnerability is classified as an exploitable weakness, where knowledge exists of the threats that could exploit those weaknesses. The level of vulnerability is determined by assessing what harm could be caused by an attack on each weakness. Thus to reinforce this set of statements:

- A weakness within a system offers a potential point of attack for a threat agent.
- Viable attacks will not necessarily be possible at all weaknesses.
- Only where an attack could realistically be mounted can that be considered to be a vulnerability.

Determining the viability of an attack the analyst has to consider the tools the attacker, Eve, has at her disposal and this is calculated as the weighted summation of the following factors which are further defined in ETSI TS 102 165-1 [i.2] as metrics:

- System knowledge:
  - Refers to the level of knowledge of the asset available to Eve and ranges from public information such as can be found on the internet , through to access to c**ritical** information about the asset (e.g. knowledge that is known by only a few individuals, access to which is very tightly controlled on a strict need to know basis and individual undertaking).

NOTE: As stated in ETSI TS 102 165-1 [i.2] open source software is an example of asset design or implementation that is wholly in the public domain, however the level of risk represented by open source is tempered by the level of vulnerability it exhibits. There is some evidence that open source software is open to greater scrutiny to resolve errors.

- Time:
  - The amount of time Eve has to be able to access the system to identify that a particular, potential, weakness may exist, then to develop an attack method (threat agent) and to sustain effort required to mount the attack.
- Expertise:
  - This metric refers to the level of generic knowledge of the underlying principles, product type or attack methods (e.g. Internet protocols, Unix operating systems, buffer overflows) required to attack the system. The levels of expertise to be applied within this factor range from laymen through to expert, and includes teams of experts working together.
- Opportunity:
  - The metric for opportunity addresses the issue that identification or exploitation of a vulnerability may require considerable amounts of access to an asset, it is noted though that spending more time around the to be attacked asset may increase the likelihood of detection. Some attack methods may require considerable effort off-line, and only brief access to the asset to exploit. Access may also need to be continuous, or over a number of sessions.

EXAMPLE: The Stuxnet attack on a particular make and model of industrial equipment required direct physical access to the equipment. In contrast the Mirai botnet identified equipment without the attacker needing direct physical access, nor did the attacker have any knowledge of which specific IoT devices were exploited.

- Equipment:
  - The metric for equipment addresses the form of equipment that Eve will need to exploit and range from standard (i.e. off the shelf with zero or minimal configuration) through to bespoke and multiple-bespoke equipment. There is an implicit link between expertise and equipment where it is often considered that an expert and motivated attacker is more likely to invest in bespoke or customised equipment.

When considering the threats and vulnerabilities of IoT devices it should be noted that many IoT devices are low cost and readily available. With such a device profile a motivated attacker may choose to spend more time and money to establish an attack in the knowledge that a single attack may impact many devices. In part there are classes of vulnerability in IoT devices resulting from re-use of connectivity and other functional capabilities across devices, hence any attack developed against a common function may be applied to any device using that common function.

A.4 Determination of risk

This addresses step 7 of the method/process.

Risk is defined in ETSI standards as the product of the likelihood of an attack (occurrence likelihood) and the impact of the attack on the system.

The likelihood of a threat occurring (occurrence likelihood) may be estimated with values from 1 to 3 as explained in table 11 of ETSI TS 102 165-1 [i.2] (Occurrence likelihood). Capability and motivation are each taken into account in the calculation of likelihood. A highly motivated and capable threat agent (e.g. a nation state with a cyber division) will be able to attack successfully even if the vulnerability rating is "beyond high" which results in a likelihood evaluation of possible.

For an IoT device the calculation is exactly the same as for any other ICT device, system or service.

A.5 Monitoring of threat level

Security, and the maintenance of a particular level of security, is a continuous process. Any change in the internal or external environment requires a re-evaluation of the effectiveness of the countermeasures and of the presented risk.

EXAMPLE: If a change in technology or a breakthrough in mathematics or number theory invalidates the claims of a problem to be hard then any assertions based on that claim have to be re-assessed and a new measure put in place to re-establish the security claim.

Effective security is founded on diligent observation and countering strategies against Eve. Thus Alice and Bob cannot afford any complacency – defeating Eve once does not imply that Eve will never attack again, or will not attack in a new way, thus Alice and Bob always have to be sufficiently agile to change their defences to keep Eve at bay.

It is important to note that Alice and Bob do not have full control over the environment and have to work across the entire supply chain and service chain to ensure that Eve can be effectively neutralised. For IoT this means adopting a number of practices to ensure awareness, to allow for code and services to be updated, and for activity of Eve to be reported and monitored across the entire eco-system.

A.6 Determination of applicable countermeasures

This addresses steps 8, 9 and 10 of the method/process.

Step 8 requires the analyst to take the risk calculation from step 7 and to then consider how to reduce the risks overall. There are a number of rules of thumb that may be followed:

- Interception on open networks can be countered by encrypting data that crosses open networks and interfaces in order to reduce the knowledge of Alice's and Bob's actions visible to Eve.
- If actions cannot be attributed to specific instances of Alice or Eve then an identity management and authentication scheme can be implemented to allow the system to give assurance that only legitimate parties are in the system.
- If there is a risk of data being manipulated a number of data integrity schemes can be put in place that restrict the likelihood of Eve manipulating data to the detriment of Alice or Bob.

Depending on where in the system implementation the security analysis is performed strategies for countermeasures can take one of two paths:

- Redesign to eliminate the source of risk.
- Addition of specific additional services and assets to reduce the likelihood of Eve attacking the system.

Once an initial set of countermeasures has been identified it is strongly recommended to perform a cost-benefit analysis to refine the countermeasure set (step 9).

EXAMPLE: If an encryption countermeasure is recommended the system then has to manage keys, almost certainly will need to manage identities, and deal with more data. All these add new assets to the system and may require changes in both hardware and software to address the provision of the countermeasure. If keys and algorithms are to be stored most security experts will recommend, if not mandate, a hardware trust anchor to be installed and from that to implement roots of trust for storage, for authentication, to accelerate the encryption, such that there is a clear boundary in the system where data is safe and where it is not (this is a concept often referred to as the Red-Black boundary in secure systems).

The final, at least until the active monitoring phase begins, is the detail implementation of the countermeasures.

A.7 Revision, verification and validation

Quite simply having performed an initial TVRA it has to become a normal business and social process in which all the previous steps are redone. This is particularly important in tracking changes to either the system or to the environment in which it operates.

Annex B:
Applying best practices to IoT security

The consumer IoT document from TC CYBER provides basic guidance for organizations involved in the development and manufacturing of consumer IoT devices ETSI TS 103 645 [i.10]. It is noted that the reference contains a large number of normative statements which do not apply as mandates in the present document. Thus the present document considers the impact of each of the best practices in terms of how a student is expected to learn its application.

- Not having universal default passwords:
  - The concern here is that Eve can easily identify any default password and thus gain authorised or higher privilege in maintaining the IoT device. The practical approach is to ensure, as much as is reasonable, that every instance of Alice has a unique set of credentials (username and password) to access any higher privilege capability of the IoT device.
- Implementing means to manage reports vulnerabilities:
  - The concern that is being addressed is that if any entity in the supply chain (e.g. user, manufacturer) finds a vulnerability then in order to fix it there has to be a means to get information back to the responsible user (i.e. most likely the manufacturer).
- Keep software updated:
  - One of the inevitabilities of software is that over time it will be attacked and need to be updated to mitigate the attack. Mechanisms to ensure that software updates are only possible from an authorised entity and where the change can be verified as legitimate will need to be provided. There are risks and consequences of this including that the software architecture and its supporting platform has to be designed in such a way that there is sufficient overhead in memory and processing capacity to allow for safe update.
  - If however there has been a design decision that inhibits software update then an equivalent hardware replacement (device replacement) solution to mitigate attack has to be designed.
- Securely store credentials and security-sensitive data.
- Communicate securely:
  - In simple terms this means send data only to known, authenticated, parties who have the authority to receive the data. In order to protect from eavesdropping on wireless links data should be encrypted, and also encrypted if the network itself is untrusted. In general any transmission via a wireless link should be treated as untrusted.
- Minimize exposed attack surfaces:
  - Very simply this means that any unused software and network ports should be closed and the design should ensure that they cannot be opened. Similarly hardware should not unnecessarily expose access to attack (e.g. open serial access, ports or test points).
- Ensure software integrity:
  - Software integrity is complex in many respects but what is often considered here is that the user (or their device) is able to verify that the software provided is the same as the software that the supplier released. Most often this requires.
- Ensure that personal data is protected
  - This is a consequence of good design but has further weight behind it as a design mandate thanks to the GDPR. Quite simply if an IoT device can be made to reveal personal data to an unauthorised entity there are considerable penalties that may be enforced.
- Make systems resilient to outages.

- Examine system telemetry data:
  - If telemetry data is collected from IoT devices and services, such as usage and measurement data, it should be examined for security anomalies.
  - If telemetry data is collected from IoT devices and services, the processing of personal data should be kept to a minimum and such data should be anonymized.
  - If telemetry data is collected from IoT devices and services, consumers need to be provided with information on what telemetry data is collected and the reasons for this (this is covered in greater detail in the best practices for privacy protection described in ETSI TR 103 591 [i.6]).
- Make it easy for consumers to delete personal data:
  - Devices and services should be configured such that personal data can easily be removed from them when there is a transfer of ownership, when the consumer wishes to delete it, when the consumer wishes to remove a service from the device and/or when the consumer wishes to dispose of the device.
  - Consumers should be given clear instructions on how to delete their personal data.
  - Consumers should be provided with clear confirmation that personal data has been deleted from services, devices and applications.
- Make installation and maintenance of devices easy:
  - Installation and maintenance of IoT devices should employ minimal steps and should follow security best practice on usability. Consumers should also be provided with guidance on how to securely set up their device
- Validate input data:
  - Data input via user interfaces and transferred via application programming interfaces (APIs) or between networks in services and devices has to be validated before being acted upon (further details of this are given in Annex F as part of safe programming practice).

In addition to the guidance given in ETSI TS 103 645 [i.10] there are further, more generic, guidelines on the application of security controls defined in ETSI TR 103 305-1 [i.13].

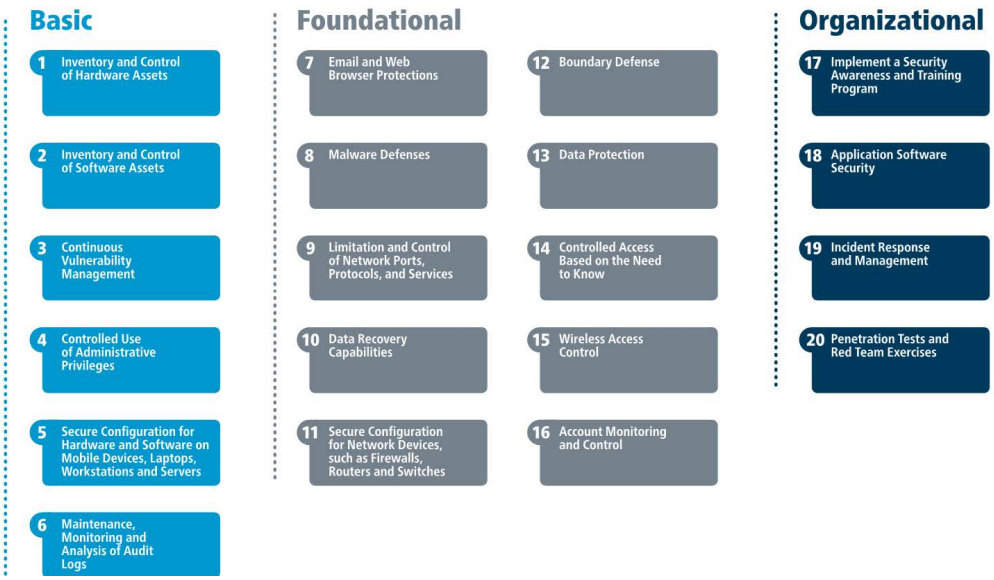


Figure B.1: Organization of the Critical Security Controls (from ETSI TR 103 305-1 [i.13])

ETSI TR 103 305-1 [i.13] identifies controls, and their associated sub-controls, in part by their core function (one of Detect, Identify, Respond, Protect) and by the asset type (one of Device, Application, Data, Network, User). In applying the security controls from ETSI TR 103 305-1 [i.13] the nature of the deployment environment has a significant role in determining how to select the particular controls to apply (across the entire eco-system all controls should be applied).

Annex C:
Cryptographic security basics

C.1 Role of cryptography in security

The primary role of cryptography is to provide the locks and keys to tighten a robust system. Of itself cryptography cannot make a system secure.

EXAMPLE: If a house is built such that the door is the only way in then securing the door with an effective lock and key will be a reasonable approach to secure against unauthorised access, but if the door is not the only way in then an attacker may simply bypass the door and enter through another unsecured entry point (e.g. an unlocked window, an incomplete wall).

The capabilities of cryptography enforce aspects of the CIA paradigm:

- Authentication, using both symmetric and asymmetric modes, and through asymmetric trust models provide assertions through the agency of Trusted Third Parties.
- Integrity proof and validation using cryptographic hashing routines (that output a fixed length digest of any arbitrary long input and where any small change in input gives a large change in the observed output).
- Confidentiality, where cryptographic encryption assures that only parties to the key can access data.

Supporting all of cryptography is key management and underpinning that is the concept of crypto-agility: The means to update the keys and algorithms used to protect assets.

C.2 Historic roots of cryptography

The purpose of cryptography and its historic roots can be considered in the study of various forms of encryption. The use of encryption is to ensure the confidentiality of a document that is sent from Alice to Bob in such a way that Alice and Bob can access the true content and where Eve, the adversary, cannot. In order for encryption to be a success the method has to be above the skillset of Eve to easily overcome. In its earliest incarnation where very few people could read, and when even spelling, syntax and semantic structure of written language was idiosyncratic, a simple substitution cipher, in which each letter in the alphabet was replaced (substituted) for another, was sufficient to thwart Eve:

EXAMPLE 1: The ciphertext "VHFXU LWBAR UNVKR SVFDQ EHIXQ" is trivial to break as every letter in the (English) alphabet is simply substituted by a letter a fixed distance down the alphabet to give "SECURITY WORKSHOPS CAN BE FUN", the Caesar cipher with a key of 3 meaning A=D, B=E, C=F and so forth.

The Caesar cipher was successful in large part as important messages were transmitted by trusted messengers (this is fundamental tool of Communications Security), who if caught (captured) were prepared to destroy the message, and even if Eve could get the message, the level of literacy at the time was sufficient to make even such a simple method of encryption very successful.

As Eve became increasingly literate, and as Eve also recognised the value of intercepting messages, it was clear to Alice and Bob that simple substitution would no longer be sufficient, that led to a number of developments, first toward polyalphabetic ciphers and then to transposition. The Vigenère cipher, a particular form of polyalphabetic cipher, was published in 1586 and remained unbroken until 1863. A polyalphabetic cipher is performed almost identically to a simple Caesar cipher but whereas in the simple case the substitution is constant over the entire plain text (i.e. every occurrence of A is replaced by D) in the Vigenère cipher the cipher is "keyed" where the key informs the order in which substitution occurs.

EXAMPLE 2: Using a substitution matrix and a keyword a plaintext "ATTACKATDAWN" is keyed by the word "LEMON" which is itself repeated to be the same length at the plaintext, then substitution of the first letter "A" is determined by the "L" column, the second letter "T" by the "E" column, the third letter "T" by the "M" column and so forth, resulting in the ciphertext "LXFOPVEFRNHR". Without knowledge of the keyword decryption is extremely difficult (cannot be broken by simple examination as can be done for a Caesar cipher).

Substitution, particularly using poly-alphabetic ciphers, can be powerful. However a second major approach in the form of transposition was introduced as early as in Ancient Greece with the Scytale cipher. A simple example of a transposition cipher is to feed the plaintext into a matrix, writing horizontally across the rows, and reading data for transmission from the columns. As with polyalphabetic ciphers transposition ciphers can be "keyed" to make them more difficult to break.

EXAMPLE 3: A 6-column columnar cipher is keyed using the keyword "ZEBRAS", where the keyword determines the order in which columns are read out, taking a plain text of "thecowsjumpedoverthemoon" results in a cipher text of "oproeuvmhjoecmeowetntsdh".

All of these keyed examples are "symmetric", the same key is used in both encryption and decryption, as is the same algorithm. These principles of substitution and transposition are still in use today, the AES algorithm uses both techniques in rounds of substitution, transposition, transposition, all keyed to achieve very strong cryptographic encryption.

However symmetric encryption does have issues as a single key is used for both operations and if data needs to be shared with multiple parties then so does the secret key. Claude Shannon proved in 1949 that a one-time pad, in which the key is both truly random and the same length as the plain text exhibited a property he termed perfect secrecy, that is, the ciphertext C gives absolutely no additional information about the plaintext. Extrapolating this is the basis of many modern communications algorithms where the key and a random element are used to generate a long key sequence equal in length to the plain text.

EXAMPLE 4: In the TETRA encryption scheme described in ETSI EN 300 392-7 [i.15] for simple voice communication a new key stream segment 432 bits in length is generated for use in encrypting every transmission packet of 432 bits. In essence the key stream acts as if it were a one-time-pad for its associated plain text. Similar schemes are used in GSM, DECT and have been extended into 3G, 4G and likely into 5G systems.

A though experiment by Ellis of the UK's GCHQ in the early 1970s posited the possibility of non-secret digital encryption, which was expanded upon by his colleague Clifford Cocks in 1973 as the foundation of what is publicly better known as the RSA encryption scheme, and then in quick sequence with Malcolm Williamson they developed a scheme for key exchange which has become better known as the Diffie Hellman Merkle key exchange algorithm. The UK discoveries were kept classified by the UK Government policy regarding such technologies that was in place at the time but later made public to highlight the role of these experts in developing the field.

The work of Ellis, Cocks, Williamson, and their counterparts in the USA, (Diffie, Hellman, Merkle, Rivest, Shamir and Adleman), are rooted in mathematically hard problems. These problems are factorisation, and the discrete logarithm problem, which allow cryptographic keys to be split such that a public part can be freely distributed in the knowledge that there is no way, that is feasible, to determine the matching secret (or private) part of the key. An operation carried out by the public part can only be undone by the private part, hence if data is encrypted by the public key it can only be decrypted by the private key.

Since its discovery and development in the 1970s public key cryptography has become the foundation of the internet connected society, and as a direct consequence to the viability of a secure IoT.

Looking to the future it is noted that there are threats to asymmetric encryption from developments in Quantum Computing which when realised nullify the hard problems currently offering security in asymmetric cryptography. In attempting to address this concern a number of groups including ETSI are working to provide a path to Quantum Safe Cryptography that will ensure that systems and assets will remain protected.

C.3 Relationship identification to pre-select cryptographic architecture

Relationship cardinality is a useful pointer to the required cryptographic architecture. A pre-established one-to-one relationship suggests the use of a symmetric key to secure communication between the parties. This is the normal relationship in networked telecommunications through a trusted central entity, thus in 2/3/4G cellular network where Alice (represented by the terminal (smart(phone)) has a trusted relationship with the home network (Bob).

For the many-to-one, one-to-many case, asymmetric keying is the most natural architecture.

EXAMPLE 1: In a retail environment many instances of Alice wish to send data confidentially to Bob, the retailer, thus Bob shares his public key with each instance of Alice, Alice can then send complete details of her order to Bob with confidence that only Bob will see the content.

For some asymmetric options such as those exemplified by Functional Cryptography (which includes Attribute Based Cryptography and Identity Based Cryptography) a single ciphertext may be selectively opened.

EXAMPLE 2: In Identity Based Encryption the public key is a common public attribute such as an email address, with Attribute Based Encryption the key is a combination of public attributes of Alice that may include location, role, time of day.

C.4 Core cryptographic modes

When encrypting data the way in which the key is used has an impact on the security offered by the key. As indicated above perfect secrecy is offered using a one-time pad and in practical systems where the size of the encrypted data is not always known in advance it is considered practical to add a time variant parameter to modify the key every time it is used. The byword is that a key should not be used to encrypt data that is longer than the key, i.e. thus a 128 bit key is good for encrypting 128 bits of data, but if the same key is used without modification to encrypt multiple data elements an attacker may use the data being encrypted to derive knowledge of the key.

Annex D:
Secure configuration of IoT devices

Securely configuring an IoT device is not dissimilar to securely configuring and other ICT device. The base guidelines and best practice outlined in Annex B are extended slightly here but the training advice for best practice is largely identical. Additional consideration is given in the present annex to the GSMA guideline [i.12] (GSMA IoT security Assessment Checklist) and should be considered alongside the application of the Security Controls defined in ETSI TS 103 305-1 [i.13] and the advice given in ETSI TS 103 645 [i.10] (see also Annex B).

A secure configuration means that all available security features are switched on and where the device actively defends every data element that is transmitted (encryption, integrity check added) and verifies that every data element received has come from a known source (i.e. identified and authenticated) without any modification (i.e. the integrity of the data is confirmed).

NOTE: In the attached pack of presentation material given as an electronic attachment to the present document the content addressing secure configuration is enveloped to the material on best practice.

Annex E:
Secure operation of IoT devices

As for Annex D secure operation of IoT devices is not dissimilar to secure operation of any other ICT device. The base guidelines and best practice outlined in Annex B are extended slightly here but the training advice for best practice is largely identical. Additional consideration to the GSMA guideline [i.12] (GSMA IoT security Assessment Checklist) should be considered alongside the application of the Security Controls defined in ETSI TS 103 305-1 [i.13] and the advice given in ETSI TS 103 645 [i.10] (see also Annex B).

NOTE: In the attached pack of presentation material given as an electronic attachment to the present document the content addressing secure operation is enveloped to the material on best practice.

Annex F:
Programming guide for secure IoT

F.1 Overview

Some programming languages are more open to allowing security errors than others. How data is passed between functions is one concern, and the interpretation of data at source and destination is often a difficulty. Programming in such a way that the likelihood of introducing errors is minimised is essential. The choice of language, compiler, operating system, even of processor all contribute to the success of safe and secure programming. At root however are the simple things:

- Document what code does, and how it does it:
  - Usually by the time a bug is found the original programmer has moved on and unless the code is fully documented nobody will be exactly sure what the code does, how it does it, and why it does it.
- Functions should be single entry and single exit:
  - Separating code into a library of functions is good practice, what is not good practice is to have more than one way of exiting the function as that is bad design in that it introduces uncertainty and can lead to code that will never be properly tested.
- Test and validate before release:
  - Most applications are complex and in an IoT networked environment may behave very differently in the field than in the lab. For example if the network is slow to respond, or fails to respond. If the user makes unexpected inputs, if the sensor that feeds data is mounted to give strange (not necessarily wrong but unanticipated) input. All of these need testing.
- Verify that out of range input is recognised and do not act on it – raise an exception and make sure that when the exception is triggered that it is acted upon:
  - Different programming languages have different ways of raising exceptions (e.g. in Swift the key words **try**, **catch** and **throw** are key to building an exception handler).
- Follow a style guide:
  - Most large organisations have a style guide for coding. The style guide will address how to distinguish variables from constants, how to structure functions, how to pass data, how to present data to the user. A good style guide will assist in achieving all the guidance above.

Examples of good practice for programming and achieving secure code are available from a large number of sources including:

- OWASP Secure Coding Practices, Quick Reference Guide (from the Open Web Application Security Project).
- The Microsoft C# programming guide and best practices (from Microsoft).

F.2 Data passing issues

Data can be passed essentially in one of 2 ways: By reference; By value. In the former all that is actually passed is a pointer to some memory location where the data is expected to be found, in the latter the actual data is passed.

F.3 Memory allocation issues

Using more memory than is required allows unexpected data to leak into and out of a system. For example if a programmer wants to store 10 characters, he can (using C) use the code:

```
char *ptr = (char *) malloc(10);
```

This allocates 10 bytes of storage at the memory address of ptr.

Now the programmer can attempt to store something at that location.

```
char  name[20];
memcpy (name, ptr, 20);
```

This tries to store a 20 byte record at a 10 byte location.

There are a number of ways to avoid such errors including not using *malloc()* but using *calloc()* which initialises the memory location to a known value. The programmer should also ensure that the data is going to fit:

```
If len(name) < len(ptr) …
```

F.4 Memory leakage issues

Memory leaks from most systems either in the short term, or over the long term. A memory leak is the result of allocated memory not being de-allocated when whatever is it allocated for no longer requires it. Most operating systems will perform some rudimentary cleaning of memory but programmers are habitually bad at understanding memory allocation and de-allocation and may miss scenarios where memory is not de-allocated. The example below assumes a user pressing a button to instruct a lift to go to a particular floor.

```
When button is pressed for requested_floor:
  Allocate memory for requested_floor
  Store requested_floor (variable) in memory
  if requested_floor = current_floor {
    finish
    }
    Else {
      Wait until the lift is idle
      Go to the requested_floor
      Release the memory for requested_floor
    }
```

The example above will cause a memory leak if the lift is already at the target floor (requested_floor) as the condition that is triggered to release the memory is never invoked.

Over time a memory leak will lead to no memory being available for the programme to run. The more constrained a device is on memory the more quickly this would become apparent. However one problem is that the time to recognise the problem may be extremely long (one byte at a time of unreleased memory will take some time to clog up all available memory). To fix the example the memory release instruction has to be taken out of the conditional loop as below:

```
When button is pressed for requested_floor:
  Allocate memory for requested_floor
  Store requested_floor (variable) in memory
  If requested_floor != current_floor {
      Wait until the lift is idle
      Go to the requested_floor
 }
 Release the memory for requested_floor
```

Many programming environments have support for code diagnostic tools that will identify potential memory leak errors, and some operating systems are better than others in ensuring that resources (e.g. memory) that are no longer needed are restored to the pool.

F.5 Data type issues

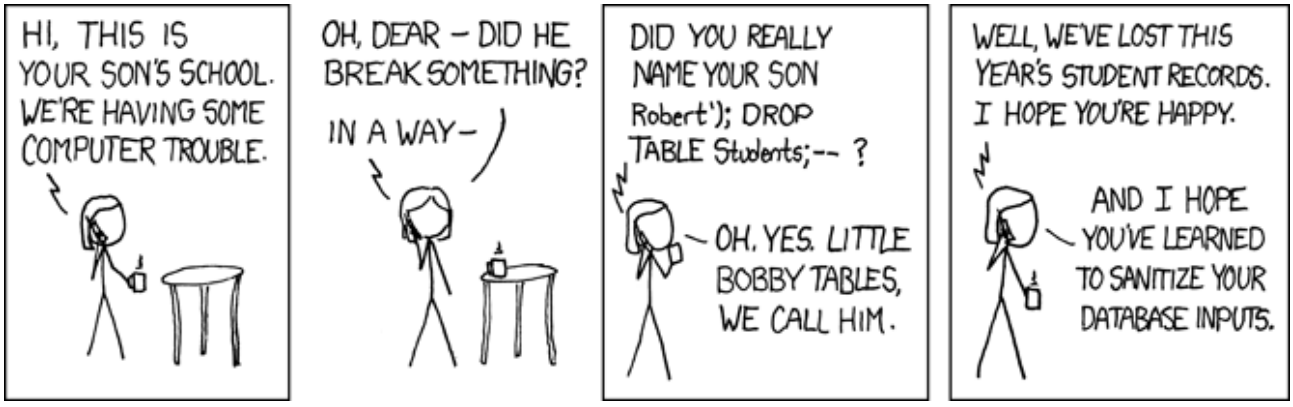
The storage of data using a type that is larger than necessary is often achieved on the fly in many programming languages. For example it is possible to declare a set of variables as integers but to explicitly cast one of them to be a floating point number if particular operations are undertaken:

```
int  var1=5, var2=7, var3;

var3 = var2/var1;
-- This operation would fail as var3 is an integer and would contain the value 1,

float var4;

var4 = (float) var2/var1;
-- This operation will give the result 1.4 as the variables not treated as integers
```

However casting of data types is potentially dangerous, for example data will be truncated when the higher data type is converted to lower, thus if a *float* is converted to *int*, data which is present after the decimal point will be lost.

F.6 SQL injection and database management issues

The database query language, Structured Query Language (SQL), can be made to introduce significant errors by injecting queries into queries. The cartoon "joke" illustrates the problem. Here the student's name is interpreted by the database server as an instruction to delete (drop) the database table "students" so when entering the student's name into the database and confirming the entry it actually results in deleting the table.



Source: https://imgs.xkcd.com/comics/exploits_of_a_mom.png

Figure F.1: Illustration of SQL injection attack

There are a number of best practices that counter most SQL injection attacks.

```
String query = "SELECT account_balance FROM user_data WHERE user_name = "
             + request.getParameter("customerName");
try {
    Statement statement = connection.createStatement( ... );
    ResultSet results = statement.executeQuery( query );
}
```

In the sample code above the "customerName" parameter is unvalidated and is simply appended to the query thus allowing an attacker to inject any SQL code they want by setting the "customerName" to some arbitrary SQL Code. Many code reuse sites contain example code that is unfortunately susceptible to these forms of attack and whilst the prevalence of such attacks is declining such methods for accessing online databases is still very common.

Countermeasures against SQL Injection centre on parsing of the input prior to execution in order to recognise and trap injected code. The following methods of defence are recommended:

- Primary Defences:
  - Prepared Statements (with Parameterized Queries):
    - Prepared statements with variable binding (also known as parameterized queries) force the developer to first define all the SQL code, and then pass in each parameter to the query later. This coding style allows the database to distinguish between code and data, regardless of what user input is supplied.
    - Prepared statements give some assurance that an attacker is not able to change the intent of a query, even if SQL commands are inserted by an attacker as parameters. This is because the parameter is used as the search parameter and it is not further parsed.
  - Stored Procedures:
    - Stored procedures are not always safe from SQL injection but if used well have the same effect as the use of parameterized queries. The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is defined and stored in the database itself, and then called from the application.
  - Whitelist Input Validation:
    - Various parts of SQL queries are not legal locations for the use of bind variables, such as the names of tables or columns, and the sort order indicator (ASC or DESC). In such situations, input validation or query redesign is the most appropriate defense. For the names of tables or columns, ideally those values come from the code, and not from user parameters.
    - But if user parameter values are used to make different for table names and column names, then the parameter values should be mapped to the legal/expected table or column names to make sure unvalidated user input does not end up in the query. Please note, this is a symptom of poor design and a full re-write should be considered if time allows.
  - Escaping All User-Supplied Input:
    - This technique should only be used as a last resort, when none of the above are feasible. Input validation is probably a better choice as this methodology is frail compared to other defenses and it cannot be guaranteed to prevent all SQL Injection in all situations.
    - This technique is to escape user input before putting it in a query. It is very database specific in its implementation. It is usually only recommended to retrofit legacy code when implementing input validation is not cost effective. Applications built from scratch, or applications requiring low risk tolerance should be built or re-written using parameterized queries, stored procedures, or some kind of Object Relational Mapper (ORM) that builds your queries for the developer.
    - This technique works like this: Each DBMS supports one or more character escaping schemes specific to certain kinds of queries. If the developer chooses to escape all user supplied input using the proper escaping scheme for the database that is in use, the DBMS will not confuse that input with SQL code written by the developer, thus avoiding any possible SQL injection vulnerabilities.
    - The OWASP Enterprise Security API (ESAPI) is a free, open source, web application security control library that makes it easier for programmers to write lower-risk applications. The ESAPI libraries are designed to make it easier for programmers to retrofit security into existing applications. The ESAPI libraries also serve as a solid foundation for new development.
- Additional Defences:
  - Least Privilege:
    - To minimize the potential damage of a successful SQL injection attack, the developer should minimize the privileges assigned to every database account in the target environment. This requires the developer to not assign DBA or admin type access rights to application accounts.
    - Start from the ground up to determine what access rights your application accounts require, rather than trying to figure out what access rights need to be taken away. Make sure that accounts that only need read access are only granted read access to the tables they need access to.

  - If an account only needs access to portions of a table, consider creating a view that limits access to that portion of the data and assigning the account access to the view instead, rather than the underlying table. Rarely, if ever, grant create or delete access to database accounts.

  - If a a policy is adopted that uses stored procedures everywhere, and do not allow application accounts to directly execute their own queries, then restrict those accounts to only be able to execute the stored procedures they need. Do not grant them any rights directly to the tables in the database.

  - SQL injection is not the only threat to your database data. Attackers can simply change the parameter values from one of the legal values they are presented with, to a value that is unauthorized for them, but the application itself might be authorized to access. As such, minimizing the privileges granted to your application will reduce the likelihood of such unauthorized access attempts, even when an attacker is not trying to use SQL injection as part of their exploit.

  - Finally steps should be taken to minimize the privileges of the operating system account that the DBMS runs under. Do not run the DBMS as root or system! Most DBMSs run out of the box with a very powerful system account. For example, MySQL runs as system on Windows by default! Change the DBMS's OS account to something more appropriate, with restricted privileges.

- Multiple DB Users:

  - The designer of web applications should not only avoid using the same owner/admin account in the web applications to connect to the database. Different DB users could be used for different web applications.

  - In general, each separate web application that requires access to the database could have a designated database user account that the web-app will use to connect to the DB. That way, the designer of the application can have good granularity in the access control, thus reducing the privileges as much as possible. Each DB user will then have SELECT access to what it needs only, and write-access as needed.

  - As an example, a login page requires read access to the username and password fields of a table, but no write access of any form (no insert, update, or delete). However, the sign-up page certainly requires insert privilege to that table; this restriction can only be enforced if these web apps use different DB users to connect to the database.

- Views:

  - SQL VIEWs can be used to further increase the granularity of access by limiting the read access to specific fields of a table or joins of tables. It could potentially have additional benefits: for example, suppose that the system is required (perhaps due to some specific legal requirements) to store the passwords of the users, instead of salted-hashed passwords. The designer could use views to compensate for this limitation; revoke all access to the table (from all DB users except the owner/admin) and create a view that outputs the hash of the password field and not the field itself. Any SQL injection attack that succeeds in stealing DB information will be restricted to stealing the hash of the passwords (could even be a keyed hash), since no DB user for any of the web applications has access to the table itself.

- Whitelist Input Validation:

  - In addition to being a primary defense when nothing else is possible (e.g. when a bind variable is not legal), input validation can also be a secondary defense used to detect unauthorized input before it is passed to the SQL query.

Annex G:
Guide to selecting a training provider

G.1 Overview

As part of best practice, it is important to also consider how security knowledge and expertise is disseminated and developed. There are a number of recognised training and certification schemes described in this annex. All of them address elements of the CIA paradigm, some with a broad-brush approach, some at a much more detailed level.

G.2 Certified Information Systems Security Professional (CISSP)

The CISSP programme addresses security from a fairly broad base across a large number of topics as follows:

- Security and Risk Management;
- Security operations;
- Identity and Access Management;
- Security engineering;
- Communications and Network Security;
- Security Assessment and Testing;
- Software development security;
- Asset security.

The overall approach has broad similarity to the approaches of things like ETSI's TVRA in that it requires knowledge of how the system can be breached with a broad knowledge of measures to protect it.

G.3 Cyber Security & Governance Certification Program

This certification program consists of 7 different certification tracks aligned based on the European e-Competence Framework (e-CF).

G.4 CompTIA Advanced Security Practitioner (CASP)

CompTIA's CASP - CompTIA Advanced Security Practitioner**,** is a vendor-neutral certification that validates IT professionals with advanced-level security skills and knowledge. This certification course covers the technical knowledge and skills required to conceptualize, design, and engineer secure solutions across complex enterprise environments. It involves applying critical thinking and judgment across a broad spectrum of security disciplines to propose and implement solutions that map to enterprise drivers, while managing risk.

G.5 Systems Security Certified Practitioner

SSCP is taught from the bottom up giving, it covers the following domains:

- Access Controls
- Security Operations and Administration

- Risk Identification, Monitoring, and Analysis
- Incident Response and Recovery
- Cryptography
- Network and Communications Security
- Systems and Application Security

G.6 DevSecOps

The growing demand for faster software delivery using the strategies of agile and DevOps, with technologies such as containers and the public cloud, has caused a rift between software production teams and security teams. Organizations are realizing that putting security reviews at the end of a production cycle is not effective, because that often causes security problems that could have been caught if security expertise had been involved from the design phase forward.

One of many solutions is to have security practices integrated into the entire software delivery cycle and this is addressed in the DevSecOps approach – Development- Security - Operation.

The method applies the same battle-tested practices of DevOps to security practices. Just as developers and operations needed to start collaborating with one another and understanding how their practices applied to the other side of the coin, security professionals also need to understand how their practices can be applied in the development and production, and later operational stages. Developers and operations need to help facilitate this change by understanding more about what security teams do.

Annex H:
Change History

Date	Version	Information about changes
09-2018	0.0.1	Outline of document structure
09-2018	0.0.2	Post STF review inclusion of common text and some change in order of text sections
10-2018 to 01-2018	003/4	Initial restructuring internally to STF
03-2019	0.0.5	Stable draft proposal
03-2019	0.0.6	Stable draft submission
05-2019	0.0.7	Final draft proposal
07-2019	0.0.9	Technical Officer review for EditHelp processing after TB approval

History

Document history		
V1.1.1	August 2019	Publication