Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Grant Agreement: 786698

D6.5 Privacy-preserving profiling: security and privacy

Work Package 6: Advanced cyber-attack detection and mitigation

Document Dissemination Level

P	Public	☒
CO	Confidential, only for members of the Consortium (including the Commission Services)	☐

Document Due Date: 30/04/2020
Document Submission Date: 30/04/2020

Co-funded by the Horizon 2020 Framework Programme of the European Union

Document Information

Deliverable number:	**D6.5**
Deliverable title:	Privacy-preserving profiling: security and privacy
Deliverable version:	1.0
Work Package number:	WP6
Work Package title:	Advanced cyber-attack detection and mitigation
Due Date of delivery:	30/04/2020
Actual date of delivery:	30/04/2020
Dissemination level:	PU
Editor(s):	Gueltoum Bendiab, Stavros Shiaeles (UOPHEC)
Contributor(s):	Gueltoum Bendiab, Stavros Shiaeles (UOPHEC) Micheal A. Skitsas (ADITESS) Sotirios Brotsis, Nicholas Kolokotronis, Konstantinos Limniotis (UOP)
Reviewer(s):	Athanasios Grigoriadis (KEMEA) Michael A. Skitsas (ADITESS)
Project name:	Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things
Project Acronym	Cyber-Trust
Project starting date:	1/5/2018
Project duration:	36 months
Rights:	Cyber-Trust Consortium

Version History

Version	Date	Beneficiary	Description
0.1	03/03/2020	UOPHEC	Table of contents proposed
0.2	10/03/2020	UOPHEC, UoP, ADITESS	Work breakdown
0.3	15/03/2020	UOPHEC, UoP, ADITESS	Partners' contribution
0.5	25/03/2020	UOPHEC, UoP, ADITESS	Partners review their act
0.8	28/04/2020	UOPHEC	Deliverable submitted for review
0.9	29/04/2020	SCORECHAIN, KEMEA	Review comments received
1.0	30/04/2020	UOPHEC	Final version ready for submission

Acronyms

ACRONYM	EXPLANATION
ABE	Attribute Based Encryption
AES	Advanced Encryption Standard
API	Application Programming Interface
ARP	Address Resolution Protocol
BGW	Ben-or-Goldwasser-Wigderson
BMR	Beaver-Micali-Rogaway
CPE	customer-provided equipment
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DHCP	Dynamic Host Configuration Protocol
DPI	Deep Packet Inspection
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
FHE	Fully homomorphic (FHE) encryption
GC	Garbled Circuit
HE	Homomorphic Encryption
HIGHT	High security and light weIGH
HMAC	Keyed-Hash Message Authentication Code
ID	Identifier
IFP	Integer Factorization Problem
iiRS	intelligent intrusion Response System
IMEI	International Mobile Equipment Identity
JNI	Java Native Interface
LAN	Local Area Network
LSOF	LiSt Open Files
MAC	Message Authentication Code
MD	Message Digest
MPC	Multiparty Computation
OS	Operating System
OT	Oblivious Transfers
PID	Process ID
PRSS	Pseudo-Random Secret-Sharing
PS	Profiling Service
RAM	Random Access Memory
RC	Ron's Cipher

RNG	Random Number Generator
SCAPI	Secure Computation API
SDA	Smart Device Agents
SFDL	Secure Function Definition Language
SFE	Secure Function Evaluation
SGA	Smart Gateway Agent
SHA	Secure Hash Algorithm
SMPC	Secure Multi-Party Computation
SSL	Secure Sockets Layer
TASTY	Tool for Automating Secure Two-partY computations
TCP	Transmission Control Protocol
TMS	Trust Management Service
TTP	Trusted Third Party
UDP	User Datagram Protocol
VIFF	Virtual Ideal Functionality Framework
VLAN	Virtual Local Area Network
WAN	Wide Area Network

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This report is a contractual deliverable within the Horizon 2020 Project Cyber-Trust: Advanced Cyber-Threat Intelligence, Detection, and Mitigation Platform for a Trusted Internet of Things. It provides a detailed description of the approach used by the Cyber-Trust project for network and IoT device profiling to identify potential security risks. In this context, the Cyber-Trust platform deployed several components to attribute and profile malicious activities and threats at the device and network level. These components are Network architecture & assets repository [A16], Smart Device Agent (SDA), Profiling Service [A17], Trust Management Service [A05] and Intelligent Intrusion Response System (iIRS, A13). The network architecture & assets repository [A16] and the Smart Device Agent (SDA) components are responsible for the acquisition of information from the end user IoT devices and gateways respectively and represent the links with the Cyber-Trust core components hosted on the service provider layer. Data from the SDA and SGA are communicated to the Profiling Service (PS) [A17], which is responsible for the storage and management of Cyber-Trust generated and collected data. The profiling service is the primary interface with the Cyber-Trust backend components, and the responsible component for gathering and collecting information from the deployed SDAs and SGAs. At the SGA level, the network architecture and assets repository [A16] component is responsible for monitoring the network traffic that flowing through the smart home gateway and collecting valuable information that can provide a much-improved view into the security health of the smart home network and detect malicious traffic patterns that might otherwise be misclassified as benign.

The aim of this deliverable is to define the obsoletely necessary data that should be collected by Cyber-Trust for further analysis and vulnerability behaviour estimation. The collected data will be used to detect active attempts to compromise devices integrity as well as abnormal payload and traffic, and initiate appropriate defensive actions (Tasks T6.2: "Device tampering detection and remediation" and T6.3: "Network attack detection and mitigation"). It will be also used by the reputation and risk assessment mechanism in the Trust Management Service (TMS) to provide a characterization of an IoT device's state (Task 5.2: "Trust establishment and risk assessment"). In such activity, privacy-preserving is a vital requirement because the collected data may contain personal and sensitive information (e.g. user identifier, hostnames, IP addresses, URLs, Payloads, etc.), which gives more chance to user privacy misuse and exploit. Therefore, in order to ensure privacy requirements, existing security mechanisms and methods for granting privacy (e.g. one-way cryptographic functions) have been explored and evaluated. Appropriates approaches will be used by Cyber-Trust platform to ensure no privacy violation occurs during data collection, and that the absolutely necessary data will only be stored in the Cyber-Trust system. This will ensure that the data to be collected and stored in the Cyber-Trust system will be compliant against legal requirements identified in tasks T3.1: "Regulatory framework analysis and personal data protection/privacy" and T3.2: "Forensic evidence collection aspects".

In addition, this deliverable explores and evaluates the appropriateness of the Secure multi-party computation (also known as secure computation, multi-party computation (MPC), or privacy-preserving computation) protocols in the constraint environment of IoT devices. Secure multiparty computation (SMPC) will be used to guarantee secure sharing of the results related to a device's assessed.

1. Introduction

The explosive growth of unsecured IoT devices creates a security gap of massive proportions that requires an expanded focus on cybersecurity. Since these devices have minimal system resources, and often include very customized and non-standard operating systems, most are not capable of accommodating management agents, leaving them invisible to traditional security management systems. Further complicating this issue is that many IoT devices have very long lifecycles and almost no security, which makes them an easy target for intruders. Exploiting such unsecured devices by hackers can lead to all kinds of potential harm and even physical damage [1]. For example, the hacker might tap into a device in an autonomous vehicle to control its driving and trigger a crash or crack a smart oven until overheats and burns the smart house down. Even a simple smart light bulb can be exploited by hackers to gain wider access to a smart home network and cause potential physical damage. IoT devices are considered not only a security threat, but also the main privacy disquiet [1] [2], as these devices gather plenty of personal data, for example, user identity, location, energy consumption, and telephone numbers. In this case, a lot of sensitive, important, and private information can be disclosed about the daily life activities of the users including using washing machines, watching TV, and leaving or returning home.

Traditional security mechanisms that involve complex software systems and relies on signature-based detection techniques, which require massive amounts of system resource to run as well as a constant series of updates to identify new threats, are not a viable solution due to the resource-constrained data-driven IoT devices. Cyber-Trust project aims to address these challenges and protects the IoT ecosystem by establishing an innovative cyber-threat intelligence gathering, detection, and mitigation platform. The proposed solution use profiling of the behaviours of IoT devices to detect active device tampering attempts and anomalies in traffic from and to them. However, profiling and monitoring such devices' behaviour and security require incorporating various layers and approaches to attribute and profile malicious activities and threats through the developing of the device and network profiling services. To this end, Cyber-Trust project developed a privacy-preserving IoT device profiling framework that dynamically and actively profiles and monitors all network-connected devices, so that any communications with high-risk devices are subject to more thorough analysis. In such activity, privacy-preserving is a vital requirement, especially with the large volume of the sensitive data generated by such insecure devices, which gives more chance to user privacy misuse and exploit by external and internal malicious entities.

Profiling IoT devices not only can gather users' private data but also can control their environments, and this fact represents the key concern. Thus, users are highly uncomfortable revealing personal data to public or private services without a well-established trust model. Therefore, the lack of any well-designed IoT-oriented privacy and security techniques will prevent user adoption to any IoT technology. In fact, users have a high interest that their privacy is protected, however, capturing and monitoring flow data and especially personal information (e.g. username, hostname, IP address, user identifier, etc.) during communication clearly violates this. To counter these privacy issues, several privacy-preserving techniques and tools have been proposed by the research commuting like hash-functions, homomorphic (HE) and Fully homomorphic (FHE) encryption methods, data perturbation, condensation techniques, etc. All these techniques can be employed within the gateway and cloud capabilities to preserve user privacy, however proper assessment needs to be made in order to check their security against various re-identification attacks (e.g. linkage attacks). Hence, this report reviews and evaluates existing cryptographic techniques and tools to find the most appropriate one that can overcome privacy challenges during the data collection and monitoring by the Cyber-Trust components.

1.1 Purpose of the document

This document aims at providing a detailed description of the approach used by the Cyber-Trust platform for IoT device profiling and monitoring along with the set of data required for describing the profile of normal behaviour (Task 6.1). The identified profile will be used for the detection of IoT device tampering attempts as well as suspicious network transactions (Task 6.2 and Task 6.3), where, every deviation from the defined profile is considered as an attack and is subject to further analysis. It also triggers the calculation of the newest device level trust score by the reputation and risk assessment mechanism implemented in the Trust Management Service (Task 5.2).

Cyber-Trust IoT platform dynamically and actively profiles and monitors all network-connected devices, where, every deviation from the defined profile trigger the gathering of valuable information that could be used as digital forensic evidence in the court of law. As data can be gathered and shared over wireless channels between the Cyber-Trust components, and able to understand and monitor the pattern and behaviour of end-user's activities, this may introduce some security and privacy issues in which user privacy can be misused or exploited. Therefore, the Cyber-Trust platform needs to ensure the protection of the individuals' data privacy by using robust privacy-preserving techniques during the data collection, as well as data sharing and management. In this context, this deliverable will explore the existing cryptographic methods for generating pseudo-identities and evaluate them against various re-identification attacks.

Further, this document aims at providing the state-of-the-art of Secure Multi-Party Computation (SMPC) protocols that can be employed by the Cyber-Trust platform to secure the sharing of gathered information between the Cyber-Trust components.

1.2 Relations with other activities in the project

This document is the second deliverable that continues the work conducted in task 6.1 (Privacy-preserving IoT device profiling), after deliverable 6.1 (State-of-the-art on profiling, detection and mitigation) which performed a thorough review of the current state-of-the-art in Cyber-Trust's profiling, detection and mitigation, namely IoT devices profiling methods, state of the art in malware detection and mitigation, as well as the quest for privacy in the Internet of Things (IoT). It also highlighted different data-related issues and threats within the Cyber-Trust project; these include identifying aspects that need to be taken care of when deploying Cyber-Trust competent and agents. More precisely, the structure and content of this deliverable are builds upon the main finding and results obtained in deliverable D6.1 and recommendations from Task 3.1 (Regulatory framework analysis and personal data protection/privacy). This will ensure that the data to be collected and stored in the Cyber-Trust system will be compliant against legal requirements.

This report intersects with the areas covered by work package WP5 of Cyber-Trust project, which include cyber-threat intelligence (CTI) gathering and sharing techniques, trust establishment and risk assessment, as well as game-theoretic security in which that the gathered and shared data should be privacy-preserved. More precisely, the output of this deliverable feeds the work conducted in task 5.2 (Trust establishment and risk assessment), which will take the device's profile that will be identified in this document as input for the reputation and risk assessment mechanism that will be developed in this task. Further, the output from deliverable D6.5 will feed the work carried out in task 6.2 (Device tampering detection and remediation) as well as task 6.3 (Network attack detection and mitigation) from work package 6. These tasks will implement the practices suggested in this document for the detection of tampering attempts as well as abnormal payload and traffic, ensure no privacy violation occurs and only the necessary data will be stored in the Cyber-Trust system. If an attack is detected, the necessary forensic evidence is collected, as identified in T3.2, and stored in the evidence repository developed in WP7.

1.3 Structure of the document

This deliverable is organized into four main sections, including the current introduction (Section 1), Conclusion and References, in order to achieve the abovementioned aim. More precisely, the rest of the document is structured as follows:

- Section 0 provides a detailed description of the profiling approach used in the Cyber-Trust platform along with a detailed description of the Cyber-Trust components involved in the IoT devices profiling, namely the network architecture & assets repository [A16], Smart Device Agent (SDA), Profiling Service [A17], Trust Management Service [A05] and Intelligent Intrusion Response System (iIRS, A13). For each component, we will explain the profiling process and provided data.

- Section 0 gives an overview of the specific collected information that will be used by the Cyber-trust platform to maintain a network profile as well as a device profile. Identified profiles will be used to detect the misbehaving nodes in real-time and mitigate the malicious behaviour including tampering attempts and abnormal payload and traffic.
- Section 0 overviews the pseudonymization techniques that have been proposed by the literature trying to find an effective and efficient method to achieve pseudonymization in IoT devises with a minimum of requirements. Appropriate technique will be used to enhance the user's privacy during the sensitive data collection, storage and processing by the Cyber-trust components. In this context, a comparative analysis will be conducted between the studied techniques based on general and technical features, especially, their efficiency against different kind of attacks like man-in-the-middle attack attacks, saturation attacks, differential attacks, etc.

2 IoT device Profiling approach overview

2.1. Overview, objectives

Generally, profiling refers to recording and analysing data that characterize personal behaviour to assess or conclude their personal interests in a specific domain or for differentiation objectives [3]. Discovered correlations and patterns may be indicative of expected future behaviour. In this context, a profile may be characterized as knowledge that allows to differentiate the relevant from the irrelevant data [4]. Profiling occurs in a diversity of domains and covers a wide variety of purposes. For instance, in e-commerce, online profiling is a key tool for companies to better understand their customer wishes, following the rule "know your customer" [5]. In modern cybersecurity, profiling is very useful in cases such as detection and mitigation of potential attacks as well as the sources of the attacks, especially with the emerging of IoT technologies. In fact, profiling IoT devices for identifying potential misbehaviour or infection by malware is critical, especially with the existing threats and the increasing number of malwares targeting such insecure devices [6], [7]. In this context, Cyber-trust project aims to provide an IoT Security Platform that dynamically and actively profiles and monitors all network-connected devices to detect anomalies in traffic from and to them and provide complete network visibility. It is considered as another level of protection against cyber-attacks and systems abuse, offering to the Cyber-Trust platform the potential to pick out new and unknown attacks, or to spot malicious activities that may be missed by the Intrusion Detection System (IDS).

In Deliverable 6.1 (State-of-the-art on profiling, detection and mitigation), we performed a thorough review of the current state-of-the-art in Cyber-Trust's profiling, detection and mitigation, namely IoT devices profiling methods, by investigating and discussing the current IoT devices profiling methods including SDA (Smart Device Agents) and could services, IoT connections and network profiling. This review showed that IoT devices operate and utilise different computing services and protocols; these include cloud platforms, network protocols, customised operating systems, and wireless connections technologies. Thereby, profiling and monitoring such devices' behaviour and security require incorporating various layers and approaches to attribute and profile malicious activities and threats through the developing of device and network profiling services. Recommendations and requirements from this report are translated by the deployment of profiling service in the Cyber-Trust platform to reliably attribute and profile malicious activities and threats at the device and network level.

The profiling service incorporates several components working together that provide important information for a given scanned service that would be saved in the network and device profiles. More information about the involved components in the profiling mechanism is provided in the following section.

2.2. Cyber-Trust components related to IoT devices profiling

IoT devices profiling involves different component in the Cyber-Trust platform. The SDA and the network architecture & assets repository [A16], which is placed in the SGA, are the two main components deployed for the monitoring and acquisition of information from the end user IoT devices and gateways respectively. The goal of these components is to extract every valuable information about the targeted device that could determine the behaviour of that device and which would be used from the other main component. One instance of each component should run on each smart gateway and smart phone respectively, in order to interact with the device and network infrastructure, and links with the Cyber-Trust core components hosted on the ISP/Cyber-Trust security provider level. The monitoring of the end user's at the SGA level is by default inactive as it enables active monitoring for all connected devices and the need to transfer exchanged traffic to the Cyber-Trust backend for Deep Packet Inspection (DPI); the user may enable or disable this option through their profile at any time after they have consented. Whereas the SDA operates in a more restrictive manner as its purpose is to receive information regarding new vulnerabilities and modes of operation from the Profiling Service [A17] and to communicate back in the occurrence of a suspicious event.

The profiling service [A17], which is running on the Cyber-Trust ISPs infrastructure, is the responsible component for gathering, storing and managing information from the deployed SDAs and the network architecture & assets repository placed in the SGAs. Necessary information about users and devices is stored within the “EndUserNetworkProfile” database. This database resides within the Profiling Service [A17] of each ISP. For security purposes, some data is hashed and stored within the Cyber-Trust DLT (on-chain). Data collected by the network architecture and assets repository [A16], and the SDA will be combined with data from the profiling service [A17], the intelligent intrusion Response System (iIRS) [A13] and the Trust Management Service [A05] for continuous monitoring of the smart home’s and mobile devices security status, the computation of possible mitigation actions to potential sophisticated cyber-attacks and trigger the calculation of the newest device level trust score.

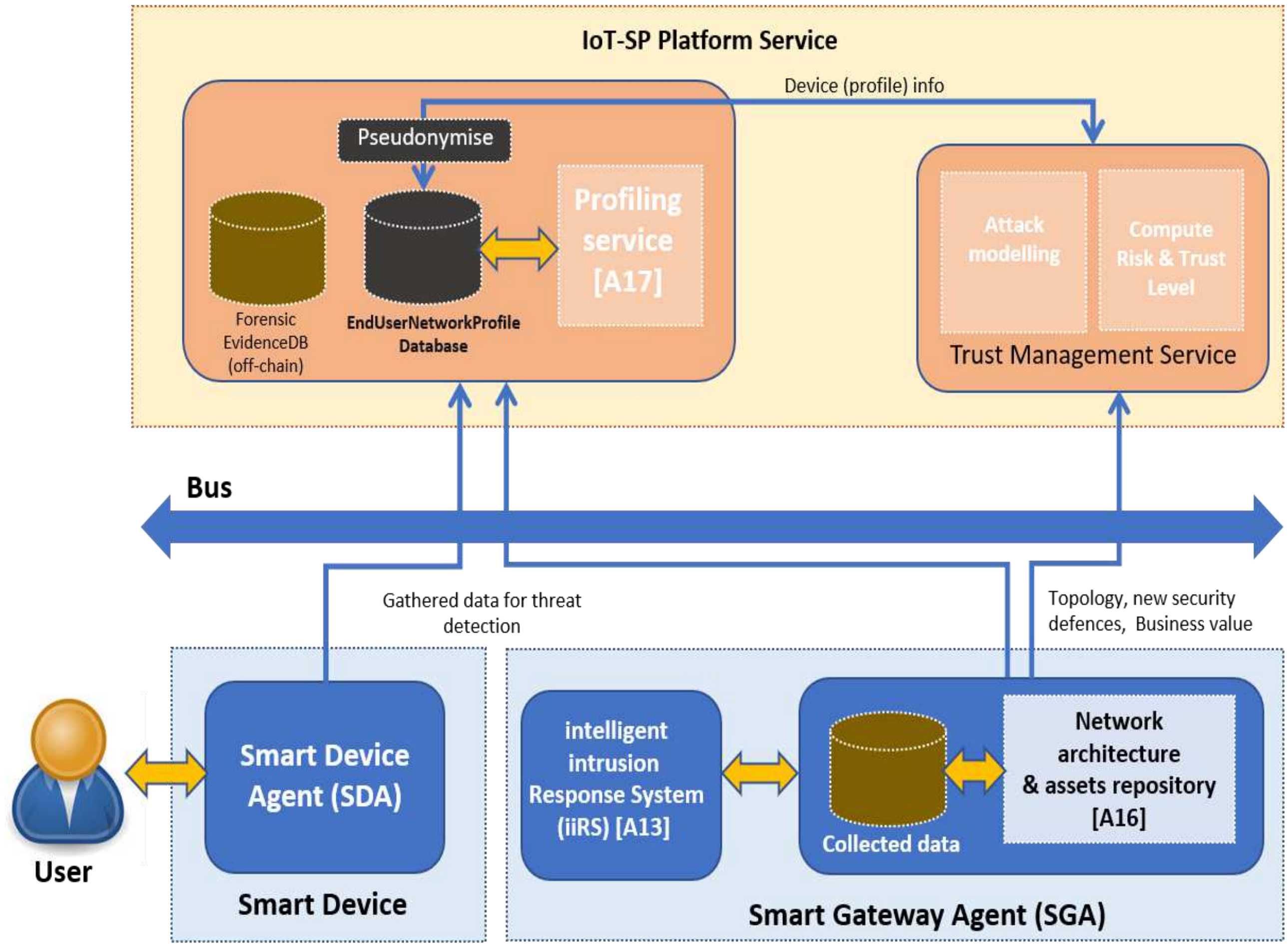

IoT-SP Platform Service
Device (profile) info
Pseudonymise
Profiling service [A17]
Forensic EvidenceDB (off-chain)
EndUserNetworkProfile Database
Attack modelling
Compute Risk & Trust Level
Trust Management Service
Bus
Gathered data for threat detection
Topology, new security defences, Business value
User
Smart Device Agent (SDA)
Smart Device
intelligent intrusion Response System (iIRS) [A13]
Collected data
Network architecture & assets repository [A16]
Smart Gateway Agent (SGA)

Figure 2.1: Cyber-Trust components related to IoT devices profiling

Figure 2.1 illustrates the relation between the Cyber-Trust components involved in IoT devices profiling. More details are provided in deliverable D4.4 (Architecture and design specifications: final). A detailed description of each component role in the profiling mechanism proposed by the Cyber-Trust platform is provided in the following subsections.

2.1.1 Network and assets repository

The network architecture & assets repository component is responsible on collecting, maintain and storing information on the network’s architecture including the topology and security defenses that are deployed at the network level, relevant device profile information, assets and their values, etc. It captures and monitors the ongoing/incoming network traffic, performs vulnerability assessment, maps hosts’ network Interfaces and provides network routing table and VLANs. As shown in Figure 2.1, the network architecture & assets repository component is also responsible for collecting and storing of device and network forensic evidence (e.g. device log files, timestamps, network data, etc.) in the Forensic Evidence database (Forensic

EvidenceDB), in order to be used as digital forensic evidence in the court of law. The gathering process is automatically performed under specific conditions. For example, with the identification of an attack.

The network architecture & assets repository is designed to be placed at the SGA and therefore one instance of this component should run on each smart home in order to interact with the network infrastructure or/and collect traffic information. The network traffic can indeed be collected from the LAN and WAN interfaces of the smart gateway and subsequently processed for storage using "NetFlow". The network infrastructure is inferred using a combination of discovery mechanisms (Nmap specifically) and querying the services on the smart gateway (from ARP and DHCP leases to VLAN and routing information). More specifically, the network architecture & assets repository provides the information described in the following table (Table 2-1).

Table 2-1: Provided information by the network architecture & assets repository

Provided Data	Description
List of network flow matrix	The network architecture & assets repository provides information about all traffic flowing between a source and destination pair in the network including the source and destination IP addresses of this flow, source and destination ports of the flow and the type of the Transport layer protocol used, Transmission Control Protocol (TCP) or User Datagram Protocol (UDP).
List of the hosts and their interface	The network architecture & assets repository also provides information about all connected devices to the network and their interfaces like the name of the device (i.e. Hostname), name of the interface, IP address and if this device is connected to Internet or not.
List of routing tables	The network architecture & assets repository fetches all the routing tables in the network and provides information about the hostname where each routing table resides, destination of each routing entry, their mask, gateway, and interface name.
List of vlans	the network architecture & assets repository also provides information about all virtual LANs on the current subnet including the name of the VLAN, IP address, mask, and default gateway.
Vulners	The network architecture & assets repository fetches the latest exploit from Vulners for a given IP address using Nmap-Vulners and nmap port scanner tools. These tools use some well-known service to provide information about vulnerabilities.

Any change in the values of the parameters described in Table 2-1 triggers an alert with the date when the change is made, and the specific parameters affected by the change. More information about this component have been provided in D6.3 (Cyber-Trust network tools).

2.1.2 Intelligent Intrusion Response System (iIRS)

The Intelligent Intrusion Response System (iIRS, A13) is responsible for the generation of the underlying attack graph, the mathematical risk assessment of the security state of the network and the choice & application of optimal remediation actions against undergoing network attacks. Two iIRS subcomponents the iIRS Attack Graph Generator (iRG) and iIRS Decision Making Engine (iRE) are the main consumers of network topology information (from the A16 component). The entire network topology is modeled by the following five aspects as illustrated in Table 2-2.

Table 2-2: Needed information for modelling the Network topology

Aspects of the Network Topology	Description
Information about each network host connectivity and identity	This aspect includes information about: • Any network interfaces, one for each different network connection of the host, and their corresponding IP addresses. • A unique hostname, to link the aforementioned IP addresses with their originating host. • A binary flag reporting whether the host is accessible from the internet (or any WAN).
Information about each and every discovered vulnerability	This aspect includes information about: • Descriptive text, allowing the iIRS Client (iRC) submodule to either interface with a human operator, or to assist with debugging and logging efforts. • The corresponding CVE identifier of the detected vulnerability. • Network connection information: the IP address of the vulnerable host, the network port, and the transport layer communications protocol (TCP, UDP, etc.)
Information about each specific network connection (i.e. the flow matrix)	This aspect includes information about: • The source IP address and network port. • The destination IP address and network port. • The transport layer protocol used for this specific connection.
Information about each specific subnetwork	This aspect includes information about: • The subnetwork's IP address range and mask (in CIDR form). • The IP address assigned to its gateway. • A unique name for the subnetwork.
Information about allowed network host interconnections	This aspect includes information about: • The destination network IP address range & mask, along with the IP address of its gateway. • The hostname and the specific interface used for the connection.

The information in Table 2-2 is obtained by the Network and Assets Repository (A16) and is used to generate the network topology model. This model is used for both the generation of the attack graph (by translating such information to Datalog facts inputted to an instance of MulVAL—the attack graph generator) and for the calculation of applicable remediation actions (using either information about the vulnerabilities themselves or by calculating the optimal network connections to disrupt).

2.1.3 Smart Device Agent (SDA)

The SDA is a software application running on the monitored device after the permission of the user. In particular, the SDA is designed to check whether the hosting device performs as intended by its manufacturer, ensures that critical OS files are uncompromised and that only secure means of communication are used. The main functionality is the acquisition of monitoring data where the synchronization with the Profiling service is performed regularly. Based on a set of rules, the monitoring data are filtered and compared with nominal values for the identification of suspicious traffic and activity. Considering the hosting OS, three different implementations have been developed within Cyber-Trust to accommodate the following classes of IoT devices: (a) Android-Based OS devices (Smartphones and Smart TVs) where the design and implementation of a Cyber-Trust Mobile App is done, (b) devices running a Linux-/Windows-based OS distributions, and (c) devices implemented to use IoT cloud Services. For each class of devices, the monitoring metrics/data have been classified in three categories: (a) the metrics related with performance such as CPU and Memory usage, (b) the metrics related with network such as open ports and, (c) the monitoring of binary files to ensure the integrity of the firmware and of the critical OS files.

Performance issues are highly related with anomalies that may appear on the devices. There are cases where the root cause of such issues is related with the user usage, the performance of applications themselves or capacity and hardware limitations. However, performance issues are symptoms of malware running on the device (abnormal behavior) that consumes hardware resources. The second class of monitoring data is related to device level network monitoring as the network interfaces are the main interface for data exchange between a device and the rest of the network including local network and internet. Monitoring of network activity at device level is mainly focused on open ports and network statistics that can be extracted from the device. The last and most important functionality of SDA is the Critical OS and Firmware Integrity monitoring of the device to ensure the operation as intended by its manufacturer. Information provided by the SDA are presented in Table 2-3.

Table 2-3: Provided information by the SDA

Provided information	Description
Performance Monitoring	The SDA monitors in real-time fields related with the performance of the system. The information is about the CPU usage, Memory and Storage of the hosted device like CPU system, RAM usage, total RAM, available RAM, and internal and external storage usage.
Critical OS Files Integrity	Based on a list of critical files that is stored in Profiling Service for each device type the following information is calculated and provided for each file: filename, path, hash (calculated value), algorithm and additional details about the signature calculation.
Device Status	The device status is a set of fields that describe the status of the hosted device. Unique identifiers of device, manufacturer, OS details are some of the data that are provided in this case. Specifically, the following information about the device is captured: Its identifier, model, manufacturer, version of the OS and built date, its serial number and IMEI (International Mobile Equipment Identity).
Installed Applications	Targeting the smart phones (Android OS) a list of installed application along with version and additional information is also monitored. Such information includes the name and version of the application, the name of the application package and the size of the application.
LSOF (LiSt Open Files) command fields	A set of fields related with currently opened files is monitored. The fields are based on the "lsof" command and include command, process ID (PID), user, size, node, path.
Network and Statistics (netstats)	Details about the network and statistics and device level are monitored through the netstats command such as local address, local port, used protocol, foreign address, foreign port, PID, etc.
Network Interface Configuration	Information related with the interface configuration through the ifconfig command is monitored.

Regardless of the type of device, the detection of abnormal behavior is based on the rules that are configured and stored in the Profiling Service and they are associated with the type of devices and user preferences. The SDA is responsible to apply the rules over the acquired data in real time and generate alerts in case of deviations. All the generated alerts are communicated with Profiling Service for storage and dissemination to the rest of Cyber-Trust components.

Details about the metrics and the modes of operation for SDA are reported in deliverable D6.2 CYBER-TRUST Device Tools.

2.1.4 Trust Management System (TMS)

The TMS plays a central role in the operation of the Cyber-Trust platform and may run on different levels of the Cyber-Trust platform architecture: (a) at ISP/Cyber-Trust security provider level, (b) at smart gateway level and (c) at the smart mobile device level. The TMS synthesizes a comprehensive trust score, taking into account the following aspects for a device **D** described in Table 2-4.

Table 2-4: Required data for computing a device score

Aspects of a device D	Description
Status	i.e. the health state of the device and the existence of vulnerabilities. Through this aspect, a partial status-based trust score SBT(D) is computed
behaviour	i.e. the observed elements of network traffic involving the device, as well as data regarding in-device activity (number of processes, disk I/O and so forth). Through this aspect, a partial behaviour-based trust score BBT(D) is computed
The risk associated with the device	i.e. the impact of any value demotion of the device, both towards the loss of assets (data and services) hosted on the device as well as towards the potential use of the device as a stepping stone for further attacks against the infrastructure, after some compromise permitting code execution. Through this aspect, a partial associated risk-based trust score ABT(D) is computed.
The peer TMS trust assessments for this device	These are sourced from trusted TMSs that are designated by the user. The individual trust assessments on device D obtained from peer TMSs are combined into a comprehensive peer TMS trust assessment score on device D as a weighted average, taking into account the trust level applicable to each contributing peer TMS.
Trust Relationships	The trust relationships between the owner of the device hosting the TMS (which coincides with the owner of the protected infrastructure, e.g. smart home, or SOHO) and the owner of the device whose trust is assessed

Initially, SBT(D), BBT(D) and ABT(D) are combined to formulate a local trust assessment for device *D*, which effectively constitutes a trust assessment based on objective evidence that is directly observable by the TMS instance. Subsequently, the local trust assessment is combined with the peer TMS trust assessment score, to produce a community-based trust assessment. The combination of the local trust assessment with the peer TMS trust assessment score is performed using an adaptive weighted average technique. Finally, the community-based trust assessment is moderated by the level of trust between the owner of the TMS and the owner of the device *D* to compute the overall trust assessment for device *D*.

The trust computation algorithm is described in detail in deliverable D5.4.

2.1.5 Profiling Service

The profiling service [A17], a centralized component that is deployed on the Cyber-Trust ISPs infrastructure, is responsible for the central storage of device profiles and correlation of monitoring data, network data, vulnerability and Cyber-Trust calculated metrics with the registered devices. The profiling service (A17) is the primary interface between the SDM and the Cyber-Trust backend components, and the responsible component for gathering and collecting information from the deployed SDAs (Smart Device Agents) and SGAs (Smart Gateway Agents). Beyond the profile of devices, the Profiling Service is extended to offer storage capabilities for the implementation of Forensic (Evidence) DB and the Patch DB. The main interface of Profiling Service supports a public REST API while the integration with Cyber-Trust Information BUS is also enabled. Using the REST API, a direct communication with the SDA is established. Through this channel, the acquisition of monitoring data, the serving of user-based and system-based configured rules and the reception of generated alerts is performed. This enables the in-direct integration of SDA with the rest Cyber-Trust components. The profiling service is integrated with the Cyber-Trust UI where the user can register, view and edit devices and their profiles. The end-user will be notified for any updates of device profiles through a developed notification mechanism.

Beyond the acquisition and storage of monitoring data and alerts from devices through the SDA, the Profiling Service is responsible for the correlation of objects (alerts, vulnerabilities, etc.) generated by other Cyber-Trust components with the registered devices. The Data Correlation Engine, part of the profiling service, is responsible to correlate data between the devices and the external sources (eVDB [A07] and Patch DB) as well as with responses and metrics from other Cyber-Trust components. The correlation with devices is mainly based on the unique identifiers that accompany a device. Each registered device has a unique identifier based on UUID generated by the profiling service. All the relationships between devices and other structures in Profiling Service are through this identifier as reference ID. For example, each record of monitoring data includes the unique identifier of the device that the SDA is running. This ID is also used by the UI for visualizing and editing information related to the device. Additionally, the IP, MAC address and CPEs related to hardware and software installed on the device are used for correlation of generated alerts by SGA and vulnerabilities.

The following table presents the necessary data collected and stored for each user, device, and smartphone.

Table 2-5: Description of the stored data for each user, device and Smart home

Entity	Description of the stored data
User	For each user, the profiling service store specific information that include his username, first name and last name, date of birth, A Boolean that indicates if a user is deleted (after deletion, data may be kept according the retention policy), his Email, gender, telephone, list of the owned devices, his role for access rights and a reference number to the Cyber-Trust AAS module.
Device	For each user, the profiling service store specific information that include the type of the device (i.e. smartphone, gateway, etc.), description of the device, information about the owner of the device and his Identifier, the type of the device owner (i.e. individual, organisation), the user of the device because sometimes user and owner are different, device information based on monitoring data where the details about the device are stored, information about the OS system, list of CPEs related with device and Reference to the smart home that device belongs.

Smart home	For each smart home, the profiling service store its name, description and reference to its owner (user). Also, configuration details for network and smart homes, small offices are also maintained under the smart home entity.

More technical details about the technology, REST API documentation and data models can be found in deliverable D6.2 CYBER-TRUST Device Tools.

As we already mentioned, the device profile is mainly developed by the information coming from the SDA [A12] and Network and assets repository [A16] and is stored and disseminated to Cyber-Trust components through Profiling Service. Two different workflows (use cases) have been defined in D2.3 and they are describing the process of profile generation.

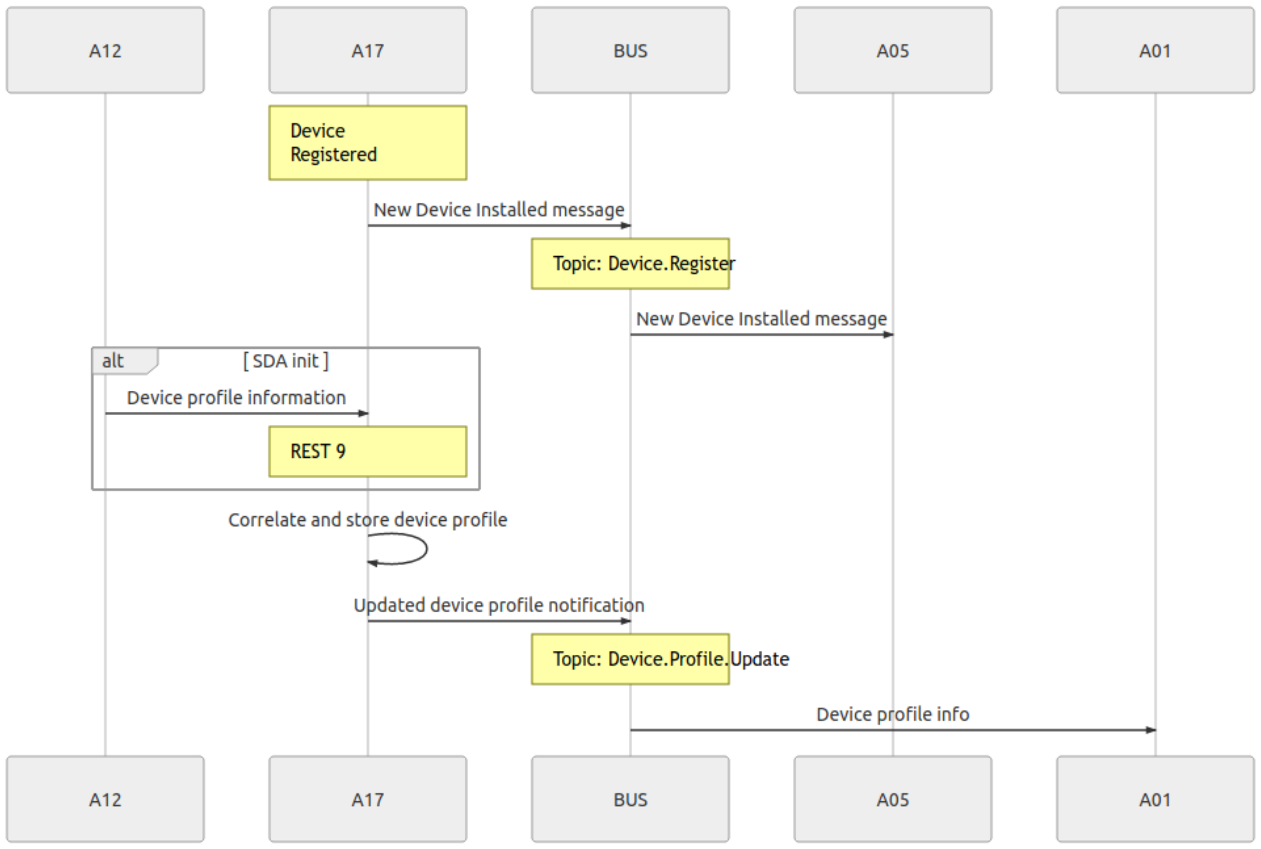

A12
A17
BUS
A05
A01
Device Registered
New Device Installed message
Topic: Device.Register
New Device Installed message
alt
[SDA init]
Device profile information
REST 9
Correlate and store device profile
Updated device profile notification
Topic: Device.Profile.Update
Device profile info
A12
A17
BUS
A05
A01

Figure 2.2: UCG-10-01: Device Profiling

Figure 2.2 presents the integration flow of the profiling service with the SDA. In particular, upon a device registration and the installation of SDA to the host device, profiling information based on monitoring and device details is transmitted to the Profiling Service. This information is stored and correlated with the registered device and is available to the rest of Cyber-Trust components.

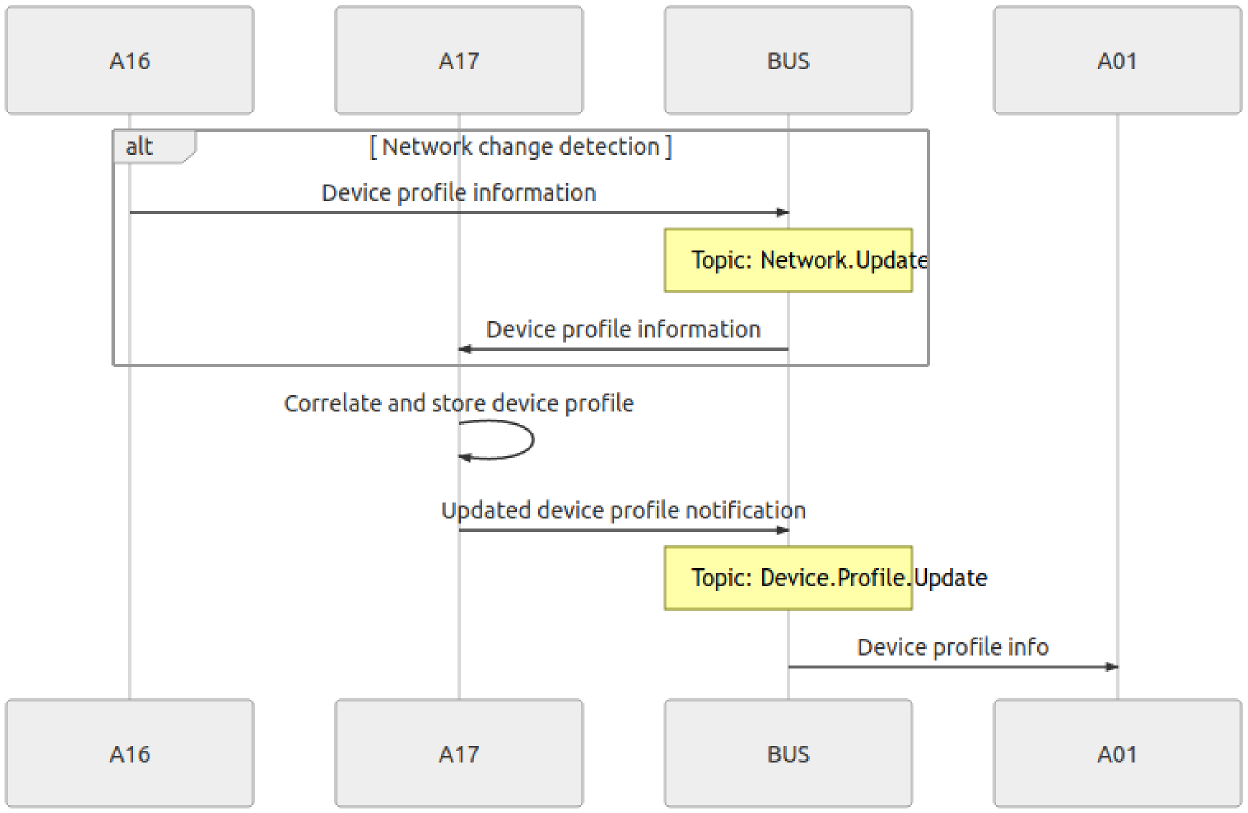

A16
A17
BUS
A01
alt
[Network change detection]
Device profile information
Topic: Network.Update
Device profile information
Correlate and store device profile
Updated device profile notification
Topic: Device.Profile.Update
Device profile info
A16
A17
BUS
A01

Figure 2.3: UCG-10-05: Gateway Network Device Profiling

Similar with SDA, information coming from A16 is also received by the Profiling Service and correlated with registered devices. Figure 2.3 presents and integration flow of such activity. It is important to mentioned, that profiling information is only stored for detected devices that are already registered to the Cyber-Trust platform.

3 Data collection and requirements

This section provides a detailed description of the specific collected information that will be used by the Cyber-Trust platform to detect the misbehaving nodes in real-time and mitigate the malicious behaviour including tampering attempts and abnormal payload and traffic. Such information may characterize the device or the network; examples include information about the integrity of a device's firmware and critical OS files, whether software patches have been installed, exposure to known vulnerabilities (in the enriched VDB), network behavioural patterns (e.g. traffic volume and protocols), and services utilisation. In this context, profiles for the devices and the network will be created and used by the Cyber-Trust Components in the detection and mitigation of malicious activities.

The device profile is the end result of the profiling and monitoring process performed from the components described in the previous sections. It contains all valuable information that is necessary for the identification and mitigation of anomalous activities. The device profile is mainly used by the TMS to compute a trust score for each device connected to the network. Whereas, the network profile, as explained in section 2.1.1, includes settings for all network connections including the topology and security defenses that are deployed at the network level, relevant device profile information, assets and their values, etc. this information is collected by capturing and monitoring ongoing/incoming network traffic and consumed by the Intelligent Intrusion Response System (iIRS) [A13] to maintain the network profile. More details are provided in the following subsections.

3.1. Device profile and trust assessment

In addition to the device profile, the TMS utilizes different types of information for each device, in order to perform a comprehensive trust assessment. The TMS operates in the broad context of the Cyber-Trust platform and sources the required information from other Cyber-Trust modules. The Cyber-Trust platform elements providing information to the TMS are depicted in Figure 3.1.

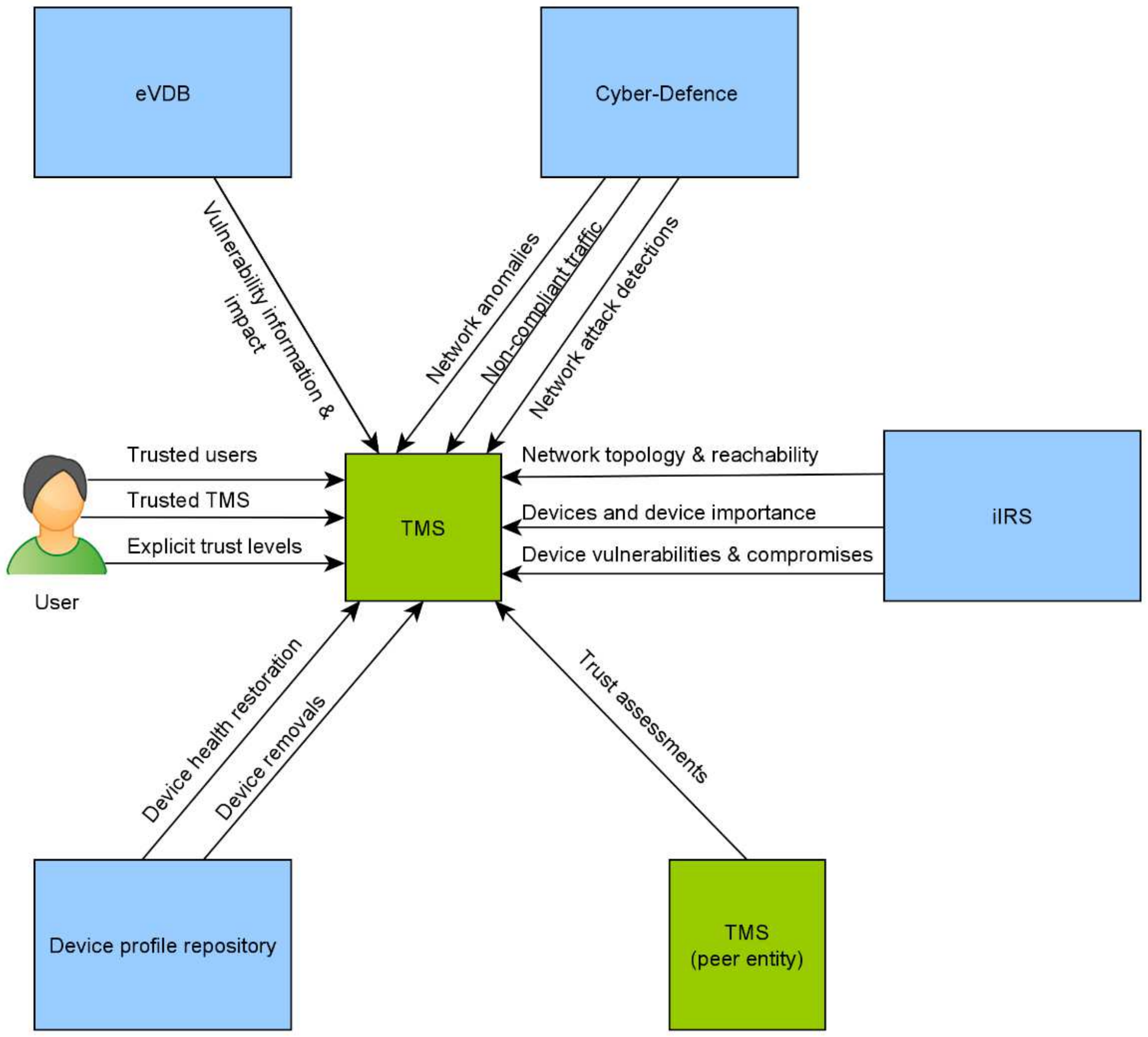

eVDB
Cyber-Defence
Vulnerability information & impact
Network anomalies
Non-compliant traffic
Network attack detections
Trusted users
Trusted TMS
Explicit trust levels
User
TMS
Network topology & reachability
Devices and device importance
Device vulnerabilities & compromises
iIRS
Device health restoration
Device removals
Trust assessments
Device profile repository
TMS
(peer entity)

Figure 3.1. Cyber-Trust platform elements providing information to the TMS

Table 3-1 provides more detail about the information sourced from other Cyber-Trust platform elements.

Table 3-1: Information sourced from other Cyber-Trust platform elements

Cyber-Trust platform elements	Provided data to the TMS
Cyber-Trust platform users	Cyber-Trust platform users provide information regarding the peer users they trust, the peer TMSs that are trusted and explicit device trust specifications. Naturally, user interaction with the TMS is mediated through an appropriate application.
The Cyber Defence module	This module provides data regarding the network anomalies detected (deviations from the nominal device and network behaviour), the non-compliant traffic (traffic flows that have not been whitelisted as "acceptable behaviour" for the device) and network attacks (primarily in the context of signature-based detection), either originating from some device or targeted against it.
The iIRS module	This module provides information regarding the devices that are in the scope of the TMS, their importance, the vulnerabilities existing

	on devices, events of device compromises, as well as network topology and reachability information. Notably, some of these information elements could be sourced from other components, especially the Device profile repository, however the iIRS module synthesises individual information elements served by the Device profile repository into more comprehensive representations, hence it was chosen to retrieve the data from the iIRS module for optimization purposes.
The eVDB module	This module provides information on the detected vulnerabilities, including their impact, underpinning the assessment of the impact that vulnerabilities may have on the trust level of the affected device.
The Device profile repository	The Device profile repository provides information on the cases that a device is removed from the system and when the device health is restored after a compromise (i.e. the malware is removed, or "clean" versions of the operating system/firmware are installed).
TMS	The TMS, acting as a trusted peer entity, provides trust assessments which are combined by the receiving TMS instance with the own device trust estimations, to synthesize a comprehensive trust score.

3.2. Collected information to maintain the Network Profile

In order to maintain a network profile, the Intelligent Intrusion Response System (iIRS) consumes the data from the subcomponents in order to produce new information or either enhances the already gathered, as its representations are considered to be more comprehensive for the association of the network characteristics with the corresponding devices. Handled information is categorized into (i) Network Topology & reachability, (ii) Device information, (iii) Decision Actions & Compromises. Extensively, the collected information can be shown in the table below:

Table 3-2: Collected information to maintain a network profile

Collected information	Description
Information regarding the Network Topology and Reachability	• IP Addresses - The addresses do not exceed the limits set by the smart environment. • Transport Protocols – (TCP, UDP, etc.) • Connectivity between the devices. • VLAN names – As given by the A16. • Device IDs – Device IDs are considered to be pseudo-identities occurring from the original device IDs given by the profiling service, using a one-way hash function to prevent/identification attacks
Information regarding the available devices in the network:	• The iIRG handles the storing of the vulnerabilities acquired from A16. Vulnerabilities are connected with information regarding their severity, description texts and general characteristics. (CVE, CVSS, etc.) • Remediation which are associated with the vulnerabilities on the network and their specific type of reference. Can be either a firewall rule or a reference.

	• Each device is also being connected with a risk score and a device importance value which is adjusted from the client and then stored. • IP Addresses - The addresses do not exceed the limits set by the smart environment. • Ports – Either as a destination or as a source. • Transport Protocols – (TCP, UDP, etc.)
Decision Actions & Compromises:	• Belief on security state of the system (for each security condition the probability that it is compromised) • Security performance trade-off parameter.

4. Review of the Privacy-preserving techniques

As mentioned in previous sections, IoT devices profiling may cause particular security and privacy risks that should be addressed in the Cyber-Trust project. In fact, collected data contain personal and sensitive information (e.g. user identifier, username, hostnames, IP addresses, URLs, Payloads, etc.) that may be attractive to cybercriminals. For instance, cybercriminals seeking to benefit financially from the theft of such data may sell the data to a third-party provider, which can then use it for commercial or even malicious purposes [8]. Further, Misuse of gathered information can lead to several privacy threats in terms of tracking, localising, personalisation discrimination and data breaches. For instance, the collected data might be used and published for purposes other than the original objective without users consent and cause significant inconvenience, or even harm to concerned users. Therefore, the privacy and integrity of collected data must be protected not only from external attackers, but also from unauthorized access attempts from inside the network (i.e. internal attacks). These objectives can be accomplished by assuring users' authentication, data confidentiality, data integrity and anonymity levels.

Data confidentiality represents a fundamental issue in IoT devices profiling scenarios, indicating the guarantee that only authorised entities can access to the collected data [9]. This require the definition of an access control mechanism to regulate and limit access to collected and stored data, based on predefined access policies. Such mechanism can be particularly effective for external attacks but is generally ineffective against internal attackers as they are likely to be authorized to access the data. Thus, integrating access control with some cryptographic primitives, such as Attribute-Based Encryption (ABE) [10] can prevent such insider threats. It successfully integrates Encryption and Access Control and is ideal for sharing secrets among groups, especially in collaborative environment. Data integrity can be compromised in several ways; through hacking, human error, whether malicious or unintentional (Insider threat), or transfer errors, including unintended alterations or data compromise during transfer from one device to another. For instance, when there is data being transmitted between two or more parties, a malicious adversary who is able to intercept the channel could alter the data. In which this could lead to a very damaging effect and sensitive data loss. In the context of the Cyber-Trust project, these data integrity compromises may be adequately prevented through access control and data encryption.

Many Privacy preserving techniques were developed, but most of them are based on anonymization of data like K anonymity [11], Data distribution [12] and Cryptographic techniques, etc. In the miscellaneous environments of IoT, exist a plethora of privacy and security issues regarding the pseudonymous communication among IoT devices. The main goal of a pseudo-anonymous communication network is to safekeep the user's communication privacy and provide anonymity of its communication identity. The difference of anonymity between the traditional communication networks and an IoT communication network is situated in the design of IoT devices. These smart devices are not equipped with high performance and high-power processors. The wireless connection that they use, their low bandwidth and their poor computational power makes them inappropriate to execute fast and efficient calculations in a secure way. Thus, it is presented below several solutions, easy to implement, in order to achieve pseudonymization in IoT devises.

4.1. Counter

A Counter provides the easiest way to achieve pseudonymization in the IoT landscape. By using a counter, the IoT devices' identities are replaced by a number selected by a monotonic integer. It is very important that the monotonic values that are created by the counter are not recurrent in order to avoid any implicity. The benefits from using a counter derive from its simplicity to provide pseudonymization without any relation to the identifiers. However, a counter can leak information regarding the order and the interactions of the pseudonymized IoT devices. In addition, this solution may face scalability issues, if a large number of IoT devices is needed to be pseudonymized, since the entire mapping table has to be stored securely in a database.

4.2. Random number generator (RNG)

RNG creates a set of unpredictable values that have an equal probability to be selected. This mechanism is quite alike to the counter, but its deference lies to the randomness of assigning a number to an IoT device. According to ENISA [13] such a mapping can be created either by a normal RNG or by a cryptographic pseudo-RNG, but it is possible to face collisions in both cases. Collisions can appear if in two or more IoT devices are assigned the same pseudonym. Although collisions may occur, the RNG method is consider more secure than the counter due to the fact that it is more difficult for an adversary to obtain information concerning the initial identity. Furthermore, both methods, counters and RNGs may face scalability or security issues, if multiple IoT devices are added in the network or the mapping table is leaked.

4.3. Cryptographic hash-functions

Hashing is a method that can be straightforwardly used to an identifier (ID) in order to create the equivalent pseudonym (PS), for example, the pseudo-identity of an IoT device can be produced as $PS = Hash(ID)$ [13]. The hash function takes as input the identifiers, which is a string of arbitrary length; and outputs a string of fixed length corresponding to the function that it is used [14]. Hash functions cannot only be used to provide pseudonyms for IoT devices, but also to ensure data accuracy[1] by satisfying the following properties [15], [13]: a) One way: If an adversary knows the pre-specified output of the function[2], it is computationally infeasible for him to find the unknown input[3]. b) Collision resistance: It is computationally infeasible for the adversary to find two distinct inputs[4] with the same hash-output. Although, the hash functions can unquestionably provide data integrity as a pseudonymization method, they are prone to attacks such as dictionary and brute force [14].

The following tables compare general and technical information of the most prominent cryptographic hash functions, namely Message Digest 4 (MD4), Message Digest 5 (MD5), Secure Hash Algorithms SHA-1, SHA-2 and SHA-3 (originally known as Keccak), Blake, Grostl, JH, Skein and Whirpool. The comparison is done based on the block size, the digest size and the word size, as well as the common attacks on the hash functions is illustrated in Table 4-1.

Table 4-1: Comparative analysis of the most prominent hash functions

Hash Functions	Block size (k)	Digest Size (n)	Keyed	Word size (bits)	Rounds/ stages	Pre-Image attack	Collision attack	Second Pre-Image attack
MD4	512	128	No	32	3	$< 2^n$	$< 2^{n/2}$	$< 2^n$
MD5	512	128	No	32	4	$< 2^n$	$< 2^{n/2}$	$< 2^n$
SHA-1	512	160	No	32	80	$< 2^n$	$< 2^{n/2}$	160- L(m)

[1] https://edps.europa.eu/sites/edp/files/publication/19-10-30_aepd-edps_paper_hash_final_en.pdf

[2] The PS in the aforementioned example or the digest of any message m

[3] The ID or any message m.

[4] ID ≠ ID’, or m ≠ m’.

SHA-2	512, 1024	160/224/ 256/ 384/512	No	32, 64	80	$< 2^n$	$< 2^{n/2}$	Min$(224, 256 - L(m))$, $256 - L(m), 384, 512 - L(m)$
SHA-3/ Keccak	1600- 2 ∗bits	160/224/ 256/ 384/512	No	64	12 + 2l - n_r to 12 + 2l - 1	$< 2^n$	$< 2^{n/2}$	$< 2^n$
Blake	512- 1024	160/224/ 256/ 384/512	Yes	32, 64	10, 14	$< 2^n$	$< 2^{n/2}$	$< 2^n$
Grostl	512- 1024	160/224/ 256/ 384/512	Yes	32, 64	10, 14	$< 2^n$	$< 2^{n/2}$	$< 2^{n-k}$
JH	512- 1024	160/224/ 256/ 384/512	No	32, 64	42	$< 2^n$	$< 2^{n/2}$	$< 2^n, < 2^{n-log2l}$
Skein	512, 1024, 2048	160/224/ 256/ 384/512	Yes	32, 64. 128	N/A	$< 2^n$	$< 2^{n/2}$	$< 2^n$
Whirpool	512	512	Yes	8	10	$< 2^n$	$< 2^{n/2}$	$< 2^n$

The comparative analysis in Table 4-1 shows that SHA-2 (Secure Hash Algorithm 2) provides a high level of security compared to other hashing functions and is today one of the hash algorithms where still no collisions have been found. SHA-2 was created because of the weakness of SHA-1. It takes the last bit group and after some bit operations, the group will be placed at the beginning. It has been shown that this step makes SHA-2 very robust against attacks. Therefore, Cyber-trust project has chosen this hashing function with bit length 256 (SHA-256) to pseudonymise sensitive data. The generated hash digest is stored in the Trust DB Admin Module [A08].

4.4. Message authentication code (MAC)

A MAC can be realized as a keyed-hush operation, in which the pseudonym is created by also adding a key. The MAC is a pseudonymization technique that provide data integrity between two parties based on a shared-private key to authenticate the disseminated information and the sender of the message. HMAC [16], [17] is considered as an attractive model and it is highly used by the most Internet protocols, since it is compatible with all the hash functions and its security is based on them. The difference is that a MAC uses a key for the compression. A MAC is a function H that takes as input the secret-key (prv_K), which is a fixed length string; and the message x, which is an arbitrary length string and outputs a fixed length string. The computation of the function (H) must be easy to be performed by the communicating parties. On the contrary, the output $H(x, prv_K)$, must be difficult to be computed by an adversary - when he knows the message x but he does not know the prv_K - even if he knows a large set of pairs $\{x_i, H(x_i, prv_K)\}$.

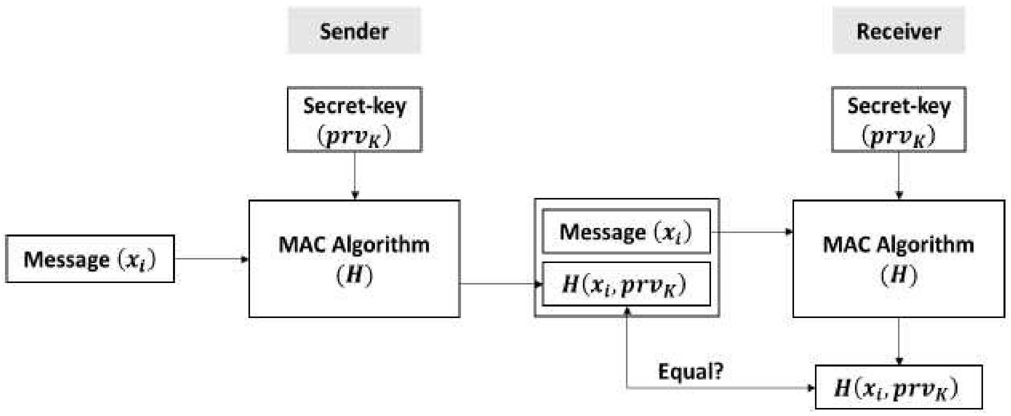


Figure 4.1: Keyed Hash Function: MAC

As shown in the Figure 4.1, the (prv_K) and the message x are given as input to the MAC. The fixed length output $H(x, prv_K)$, along with the x_i are given via an insecure channel to the receiver. When the message x_i and the MAC value $H(x_i, prv_K)$ are obtained, the receiver re-inputs the message x_i and the secret-key (prv_K) to the function (H). If the recomputed MAC value is the same with the initial, then the receiver is certain that the message x_i has been disseminated by the expected sender. Otherwise, the message x_i has been altered or its origin is not the expected.

4.5. Lightweight Encryption Algorithms (LEA)

Anonymity and security are the main concerns in the functionality of IoT devices. The ordinary cryptographic algorithms cannot be considered as the best option for the IoT landscape, since they demand from the IoT devices to possess an extensive amount of computational power and memory. For this reason, lightweight cryptography can be considered a more suitable option to be used for the communication between them. LEA are designed to use smaller internal states, short block and key sizes. Indeed, most lightweight block ciphers use only 64-bit blocks (AES is demanded a 128-bit block and a 128-bit key) [18]. The lightweight implementation usually leads smaller RAM consumption and it is good at processing smaller messages as well [18]. There are several lightweight cryptographic algorithms (both symmetric and asymmetric) that can be used for this purpose.

4.5.1. Symmetric lightweight cryptographic algorithms

A variety of symmetric lightweight cryptographic algorithms have been proposed by the research community. The Most important and suitable for low resources devices are Advanced Encryption Standard (AES), TWINE, HIGHT, PRESENT and RC5. Table 4-2 provides description of theses lightweight symmetric algorithms.

Table 4-2: Symmetric lightweight cryptographic algorithms

Symmetric lightweight cryptographic algorithm	Description
Advanced Encryption Standard (AES)	The ***AES*** cryptographic algorithm is comprised of three block ciphers, AES-128, AES-192 and AES-256, according to the length of keys that are being used. For a ciphertext to be created, the AES-128, the AES-192 and the AES-256 process the plaintext in 10, 12 and 14 encryption rounds (respectively), where each round is

	comprised by a substitution (sub byte), a permutation (shifts data rows), and a mixing (mixcoloumn) of the plaintext
TWINE [19]	TWINE is a 64-bit block cipher, based on a variant of the Feistel algorithm [20]. Which is invoked 8 times in each round. There are two versions of TWINE, namely TWINE-80 and TWINE-128, according to the bit-length key. The encryption consists of a XOR function on the sub-key and a call operation to a 4 × 4 S-box. TWINE requires a small amount of computational power to enable competent software implementations.
HIGHT [21]	is a 32-round block cipher, based on the Feistel network [20] and applicable to lightweight implementations due to the fact that the block length is 64-bits and the length of the key is 128 bits. The prestigious of this encryption algorithm is that is based on simple functions such as bitwise rotations, modulo reductions and XOR operations. In [21] the authors showed that HIGHT can be used in tiny devices and process a round of encryption per one clock cycle, which makes the algorithm much faster than the AES
PRESENT [22]	It is a SP-network [15] with an oriented permutation layer. To create a ciphertext the algorithm needs 31 rounds with block length of 64 bits and two version keys (either 80 or 128 bits). In each round a 4-bit S-box and a XOR operation are executed, the first 16 times and the later only once in order to introduce the round-key, which is applied for bit-wise permutations and substitutions
RC5 [18]	RC5 ("Ron's Cipher 5") [18], created in 1994 by Ron L. Rivest, also shows great potential for a lightweight cryptography method. It is a block cipher which has a variable block size (32, 64 or 128 bits), a variable number of rounds, and a variable key size (0 to 2,048 bits). It can thus be used to match the encryption to the capabilities of the device. If it is a low-powered device with a limited memory and a relatively small physical footprint, we could use a 32-bit block size and an 80-bit key, with just a few rounds. But we can ramp up the security if the device can cope with it, and use 128-bit block sizes and a 128-bit key. It can also be flexible, where a single change on either side can improve or reduce the requirements.

In Table 4-3, it is presented an overview concerning the differences of the aforementioned lightweight and symmetric cryptographic algorithms. These differences are based on their Code length, their key size, their structure, the number or rounds that are needed to be executed and the possible attacks that can be performed on each algorithm.

Table 4-3: Possible attacks on symmetric lightweight cryptographic algorithms [23]

Symmetric algorithm	Code length	Structure	Number of rounds	Key size	Block size	Possible attacks
AES	2606	SPN	10	128	128	Man-in-middle-attack
HEIGHT	5672	GFS	32	128	64	Saturation attack

TEA	1140	Feistel	32	128	64	Related Key attack
PRESENT	936	SPN	32	80	64	Differential attack
RC5	Not fixed	ARX	20	16	32	Differential attack

4.5.2. Asymmetric lightweight cryptographic algorithms for IoT devices:

The most well-known and generally used asymmetric algorithms are RSA and EEC (Elliptic Curve Cryptography). SSL certificates most commonly use RSA keys and the recommended size of these keys keeps increasing (e.g., from 1024 bit to 2048 bit a few years ago) to maintain sufficient cryptographic strength. However, ECC can offer the same level of cryptographic strength at much smaller key sizes - offering improved security with reduced computational requirements. Table 4-4 provides description of these two asymmetric algorithms.

Table 4-4: Asymmetric lightweight cryptographic algorithms

Asymmetric lightweight cryptographic algorithm	Description
RSA	RSA [24] took its name by its creators (Rivest-Shamir-Adleman) and it is one of the most well-known and generally used asymmetric algorithms. The length of the key - public and private, for encryption and description (respectively) - that is being used in RSA is at least 1024 bits, which makes the algorithm secure but unfortunately not quite applicable to IoT environments If the cryptographic operations are executed in the IoT devices. The security of the algorithm is based on the IFP (Integer Factorization Problem), which refers to the difficulty of finding the large prime factors (p, q) of a number $N = p * q$.
EEC (Elliptic Curve Cryptography)	The Elliptic Curve Cryptography (ECC) is modern family of public-key cryptosystems, which is based on the algebraic structures of the elliptic curves over finite fields and on the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP) [25]. ECC is considered a natural modern successor of the RSA cryptosystem, because ECC uses smaller keys and signatures than RSA for the same level of security and provides very fast key generation, fast key agreement and fast signatures. An encryption algorithm with small size keys, which are generated and create signatures really fast and without the need for the IoT devices to waste computational power, is a quite appealing method to IoT environments.

In Table 4-5, it is presented an overview concerning the differences of the RSA and the ECC cryptographic algorithms. These differences are based on their Code length, their key size and the possible attacks that can be performed on each algorithm.

Table 4-5: Possible attacks on asymmetric lightweight cryptographic algorithms [23]

Asymmetric algorithm	Code length	Key size	Possible attacks
RSA	900	1024	Module attack

ECC	8838	160	Timing attack

4.6. Secure Multiparty Computation protocols

In the era of IoT, sensitive information can be shared between smart devices. These devices can create massive volumes of private data, which can be used to make decisions on recent events or to find potential trends. Secure Multi-Party Computation (SMPC) is used in IoT devices, since two or more parties can exchange, in a secure way, private information without the presence of a TTP. With each party to possess its own private input, SMPC calculates with joint function of all the secret inputs.

A well-known example of MPC is the Millionaire's Problem. Let's assume three parties, namely: $Client_1, Client_2$ and $Client_3$. Their corresponding secret inputs that denote their income are the variables x, y, z. The objective here is to find which one has the highest income without disclosing each party's private income to the others. This problem can be solved by computing the function S, which is $S(x, y, z) = \max(x, y, z)$. When the protocol is finished, each party will have: 1) shared its input without disclosing it 2) obtained the results of the function S and 3) not obtained the secret inputs of the other parties. The security of this protocol is based on the optimal model, where a TTP receives the secret inputs from the parties, computes the function S and then sends back to each party the variable (name of the client) that corresponds to the higher income.

MPC is actually based on secret-sharing and applies methods to distribute a secret among a set of n participants. The secret can be reassembled only when a threshold of $t + 1$ shares are joint together. An example of MPC is illustrated in figure below.

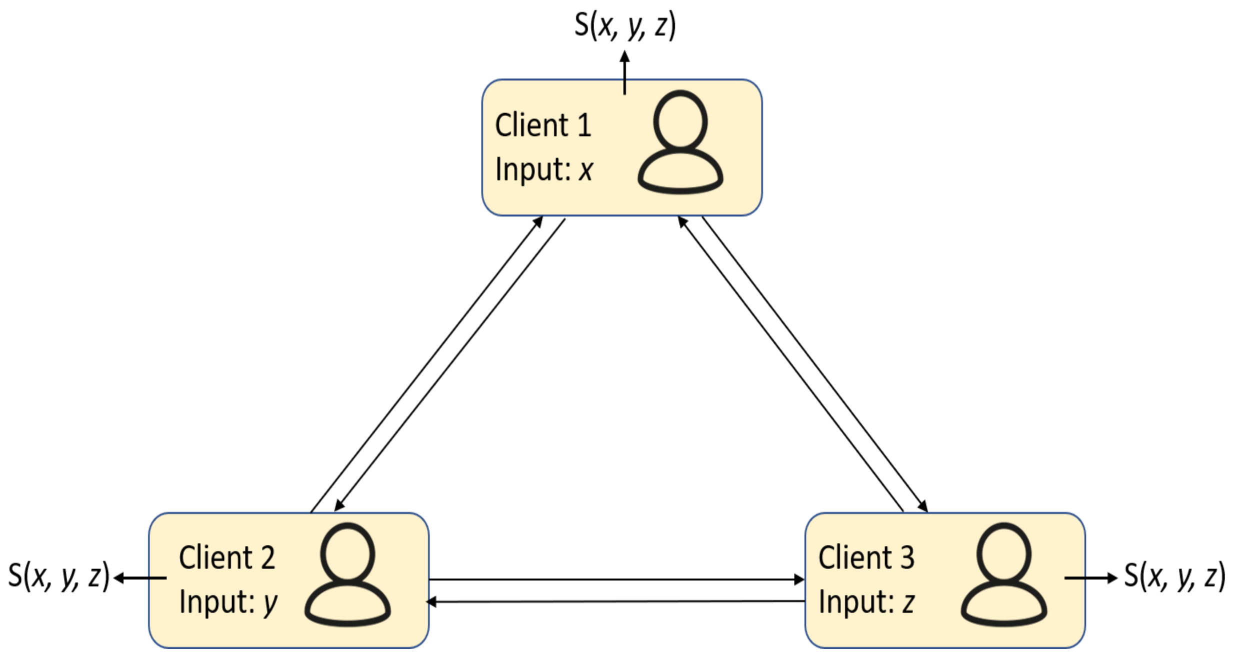


Figure 4.2: MPC without the use of a TTP

The millionaire's problem was first proposed by [26]. Let Alice and Bob be two participants, who want to know which one is the richest without revealing any secret information about their income to each other or to any trusted authority. The function S with given inputs x and y returns the name of the party who possesses the highest income, as shown in Figure 4.3. Thus, in this way privacy is preserved because Alice knows that she has more money than Bob while Bob knows that he has less money than Alice, but neither of them knows how much money the other party has.

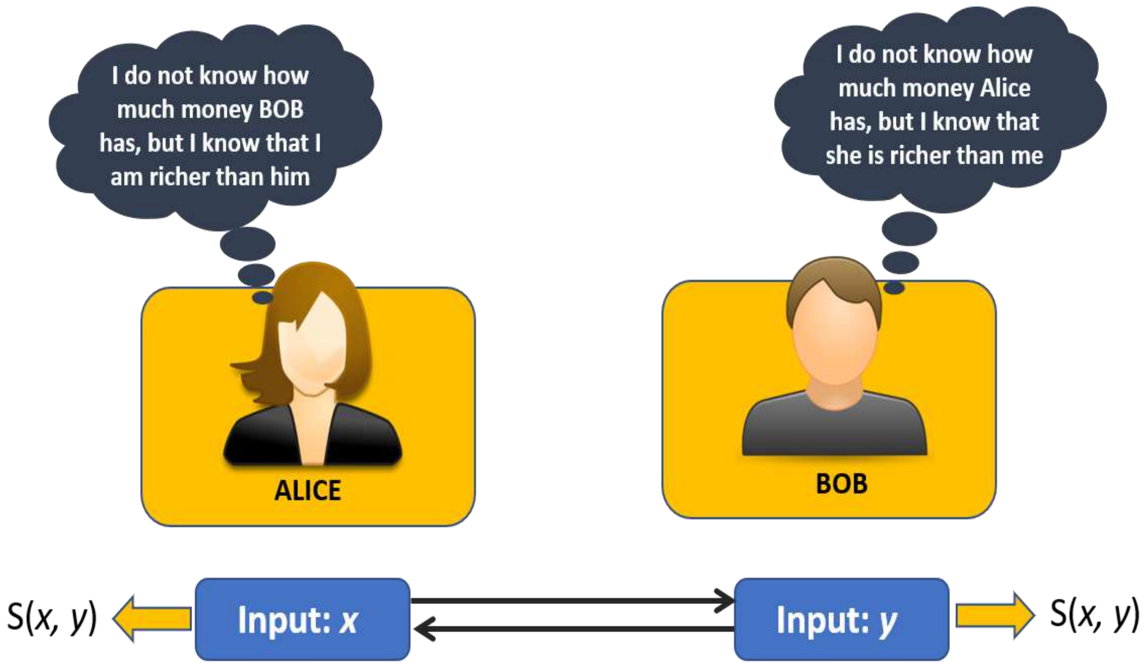

I do not know how much money BOB has, but I know that I am richer than him
I do not know how much money Alice has, but I know that she is richer than me
ALICE
BOB
S(x, y)
Input: x
Input: y
S(x, y)

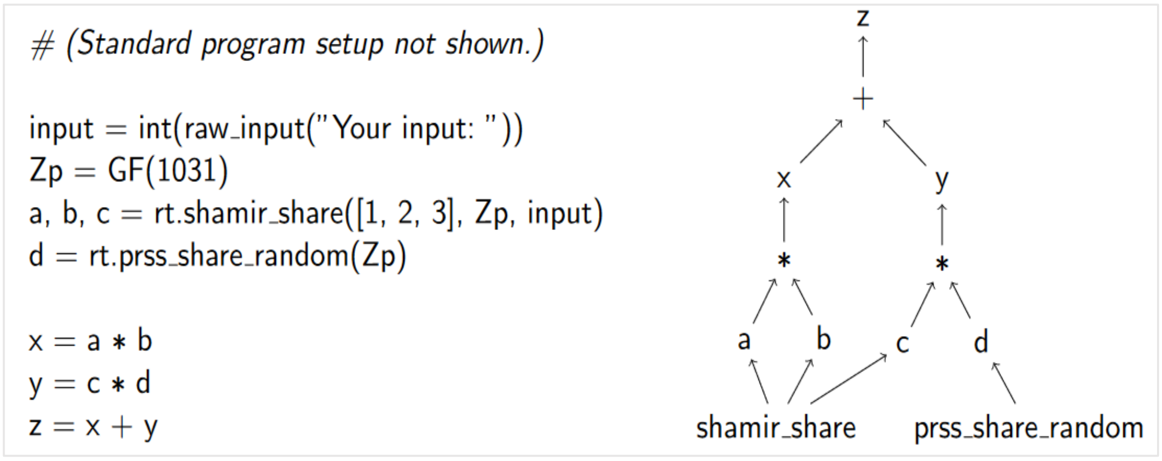
Figure 4.3: Millionaire's problem

Different frameworks are designed to enable SMPC by providing the elementary MPC functionality to allow the creation of complex applications. Several frameworks can be found for scalability, software composition, security level and performance. MPC frameworks enable users to determine a SMPC, in which the parties run a cryptographic scheme to execute a joint computation with a predetermined function without disclosing any private information regarding their inputs. Such frameworks are presented in the following subsections.

4.6.1. Virtual Ideal Functionality Framework (VIFF)

The Virtual Ideal Functionality Framework (VIFF)[5] is a library for programming cryptographic protocols. The key idea is to deliver the basis for building practical applications [27]. VIFF can be to be used by participants in real case scenarios (i.e. asynchronous networks), in which there are no limits concerning the time that a message has to be delivered. The VIFF runtime system deals with the asynchronous settings by avoiding the anticipation, except it is inquired to do so [28]. Generally, in asynchronous networks each party wait the others at the end of every round, but in VIFF there no concept of "rounds". The inherent and mutual dependencies define the order of execution, which sometimes might be unpredictable. This means that the computations stay secure when they are performed out-of-order. Asynchronous protocols benefit from this property due to the fact that the adversary can only postpone packets promptly.

[5] http://viff.dk/doc/overview.html



```
# (Standard program setup not shown.)

input = int(raw_input("Your input: "))
Zp = GF(1031)
a, b, c = rt.shamir_share([1, 2, 3], Zp, input)
d = rt.prss_share_random(Zp)

x = a * b
y = c * d
z = x + y
```

Figure 4.4: The program (left) - The expression tree (right) [28]

An example is shown in Figure 4.4 (left)[6] for $n = 3$, where a simple program starts with an input (an integer) from a user and then the field $\mathbb{Z}_{1031}$ is defined. Then all the parties join in a Shamir sharing scheme and then the three shared objects are being determined, while the fourth is created by using *pseudo-random secret-sharing* (PRSS) [29]. VIFF provides Shamir secret sharing (when $n \geq 3$) and additive secret sharing (when $n = 2$). In Figure 4.4 (right) it is presented the aforementioned execution as a tree. The arrows represent dependencies among the expressions and the outcome of the computation is the variable (z). The independent variables x and y can be also computed in parallel for efficiency reasons.

4.6.2. SHAREMIND

SHAREMIND [30] adopts secret sharing to achieve privacy-preserving computations by splitting private information among nodes (miners). Multiplication, greater-than-or-equal correlation and addition of two shared values are the computations that SHAREMIND can provide. In addition, in order to reduce the number of the required rounds to apply computations on several inputs the protocol implements vectorized computations to execute the protocol in parallel.

The SHAREMIND framework is illustrated in Figure 4.5 [30]. The data donors by sending their private shares to the miners adequately designate all their rights over their privacy to the corresponding set of miners, which use MPC to execute the algorithm. SHAREMIND can be seen as a virtual processor, which supports secure storage for the private inputs and execute privacy-preserving computations. Each miner M_i possesses a local database and a local stack. The first for persistent storage and the later for storing the results. By using *additive secret* sharing over the field $\mathbb{Z}_{2^{32}}$, the values that are included in both the stack and the database are shared to the miners. When the inputs are obtained, a data analyst can perform computations by giving instructions to the miners.

[6] rt is a Runtime object.

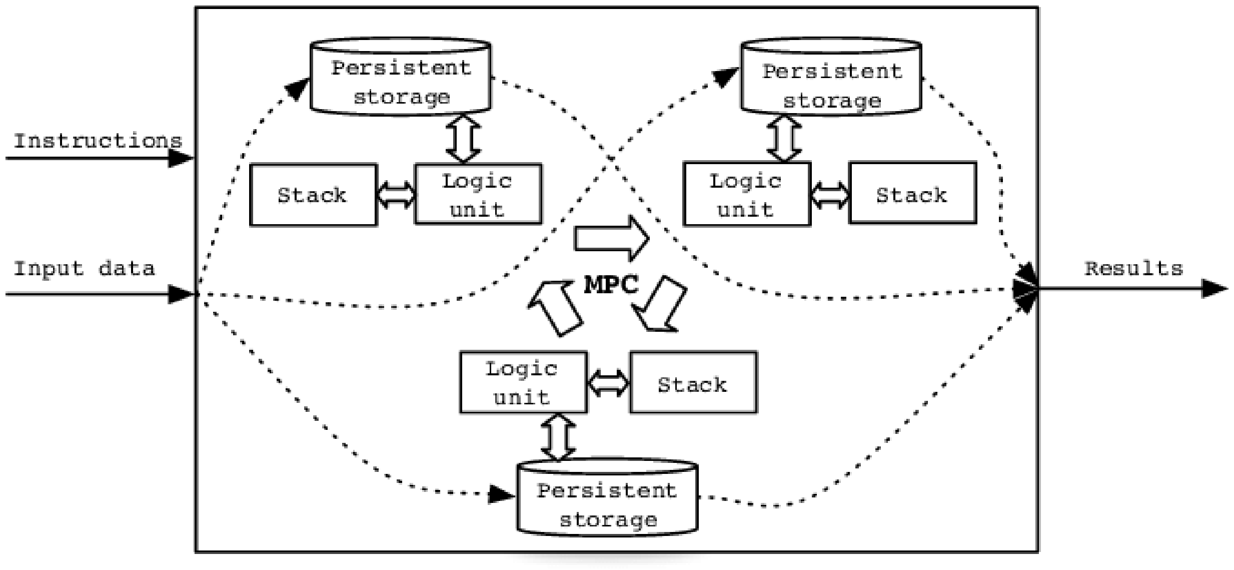


Figure 4.5: The SHAREMIND framework [30]

Each command is executed either to invoke a computing protocol or to reorder shares. The instructions, which are given by the data analyst invoke a SMPC protocol that outputs new shares to be placed on the stack. For example, a unary stack instruction S takes the top shares $[[a]]$ of the stack and sends the occurring shares $[[S(a)]]$ to the stack top [br05]. Commonly, a binary stack instruction $\otimes$ takes the top-most shares $[[a]]$, $[[b]]$ and pushes $[[a \otimes b]]$ to the stack.

4.6.3. FairplayMP

FairplayMP [31] allows the participants to run a combined computation, which emulates a TTP in order to receive the secret inputs, compute the operation, and privately notify the participants about the outputs. FairplayMP works by acquiring a configuration file concerning the participants and a description of a function. The function is compiled into a description and a distributed evaluation of a Boolean circuit is performed without the disclosure of any private data. The protocol is executed in eight constant rounds and it is a supplement of the original Fairplay system [32], while the BMR protocol works as the underlying system.

FairplayMP's structure, which is akin to [32] is described below:

- At first, the users write a program in SFDL 2.0[7]
- The aforementioned program is compiled, and it is represented as a Boolean circuit.
- Then, the users write a configuration file (config.xml) in order to describe the system's settings i.e. the security parameters (such as which port, certificate, PRG and the prime number is going to be used) and the participants' IP addresses.
- The Boolean circuit's SMPC is executed in the following steps:
    - From the Boolean circuit, a *garbled circuit* (GC) is created in accordance with the BMR [33] protocol.
    - When the output of the protocol is received from the participants, the GC is evaluated.

Assuming n players, FairplayMP is considered secure if the corrupt computation parties behave semi-honestly and they are less than $n/2$. The protocol is mostly based on the BMR and BGW protocols. Thus, assuming that these protocols are implemented in a secure way, any addition of the other parties (except

[7] https://www.cse.huji.ac.il/project/Fairplay/FairplayMP/SFDL2.0.pdf

from the computation parties) will not affect the security, since the corrupted parties will not learn anything about the parties' private inputs.

4.6.4. SCAPI

SCAPI [34] for the "Secure Computation Application Programming Interface" is an open source library[8], written in Java, for implementing secure two-party and multiparty computation protocols. It provides a reliable, efficient, and highly flexible cryptographic infrastructure that is suitable for large projects, and quick implementation. The key principles of the protocol's design are:

1. **Flexibility**: The low-level primitives of the implemented protocols in SCAPI are written in abstract terms and they are easily changeable.
2. **Extendibility**: This principle means that, in SCAPI it is possible to subsequently add new implementations, new libraries of the current protocols and primitives using wrappers.
3. **Efficiency**: SCAPI can achieve efficiency and access libraries fast by wrapping low-level primitives via the Java Native Interface (JNI).
4. **Ease of use**: Other cryptographic libraries, which are designed for specific cryptographic tasks, are not meant to be reused by others. SCAPI is easy to be used, well document, and written for secure computations.

As shown Figure 4.6, the SCAPI library is comprised of three layers, namely Basic primitives' layer**,** Non-Interactive Schemes layer and Interactive protocols layer.

1) **Basic primitives:** This layer contains the basic cryptographic primitives (e.g. hash functions, discrete log groups, pseudorandom functions, etc.). Most of the code in the lowest layer is comprised of wrapping code and provides the interface for several other libraries (Bouncy castle, Crypto++, Miracle, etc.).
2) **Non-Interactive Schemes:** This layer consists of the cryptographic schemes (such as, public or private key encryption, Message authentication codes MACs, etc.). The protocol can also support multiple cryptosystems, such as RSA [24], RSA-OAEP [35], AES [24], DSA [36], Cramer-Shoup [34], El-Gamal [37], etc.
3) **Interactive protocols:** This layer is the most powerful of the other two and consists of schemes and protocols that are extensively used for Multiparty computations (such as: Sigma protocols, GC, zero-knowledge, OT protocols, etc.).

In addition, SCAPI provides another layer, namely "the communication layer" that sets up secure channels in order to enable the multiple parties to exchange messages.

[8] https://github.com/cryptobiu/libscapi

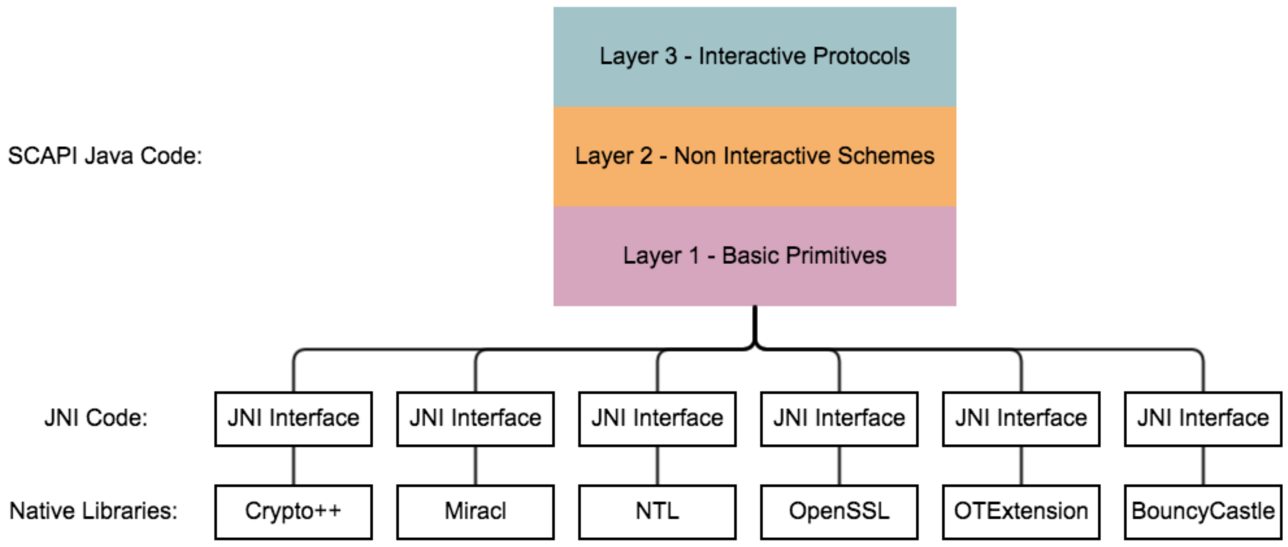


Figure 4.6: The SCAPI library[9]

4.6.5. Tasty

Tasty [38] allows two parties, which do not adequately trust each other, to execute joint computations on an arbitrary function, learn the function's output and preserve their privacy without disclosing any other information. By using HE, GCs or even the aggregation of both, TASTY provides a secure, efficient and automated two-party SFE [26] for privacy-preserving methods. TASTYL, which is created by TASTY, is implemented in Python and describes the SFE protocols as a set of functions on the private data[10]. In addition, TASTY can evaluate and compere the SFE protocols' performance. Regarding their performance, the SFE protocols aim to minimize the time that passes from the moment that the parties give their input to the function until they get their outputs (i.e. the online phase).

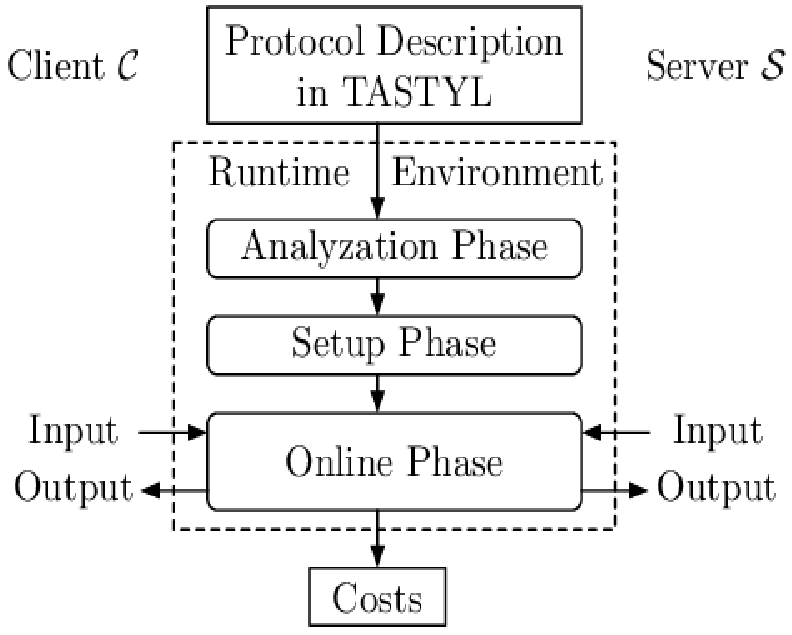


Figure 4.7: TASTY's Workflow and design [38]

[9] https://scapi.readthedocs.io/en/latest/intro.html
[10] https://github.com/encryptogroup/tasty

TASTY’s workflow and design are illustrated in Figure 4.7. Firstly, the server and the client, herein are both called users, come to an agreement regarding the SFE protocol’s description [38]. Then, the SFE is analyzed, executed and benchmarked automatically by TASTY's Runtime Environment, which is invoked by the users.

The Runtime Environment runs into three phases: In the *analyzation-phase*, in which the description of the protocol is checked for syntactical errors and it is given as input in a hash function. The output of the hash function is exchanged between the users in order to confirm both of them that they execute the identical protocol. The protocol is analyzed from the runtime environment in order to be defined the pre-computed parts (e.g. GCs and OTs). These parts are pre-computed in the *setup-phase* and they are separated from the protocol. In the final phase, the users give their inputs for computation and then the SFE protocol’s online part is executed - by using HE, OT and GC evaluation – in order to compute the corresponding output for the users. A detailed information regarding the workflows and the three aforementioned phases of each SFE protocol is costly, thus, TASTY offers a tool, which can measure such performance costs.

4.6.6. Comparison

In general, several frameworks and practical programming languages have been designed to carry out SMPC. SCAPI [34]and FairPlay [32] are based on GCs. Sharemind [30] and VIFF[11] are built on top of secret-sharing. Finally, FairPlayMP [31] combines Secret-Sharing and GCs, while TASTY [38] uses GC and HE. If the protocol requires to secure computations for only two entities, then GCs are used in the framework. Otherwise, Secret-Sharing schemes are implemented. In Table 4-6 it is illustrated a comparison of the aforementioned frameworks regarding the entities that participate in the computation, the programming language and the cryptographic schemes that each framework implements.

Table 4-6: Comparison between the aforementioned MPC frameworks [27]

Frameworks	Language	Cryptographic Schemes	Parties
VIFF	Python	Secret-Sharing	≥3
Sharemind	SecreC (C++)	Secret-Sharing	3
SCAPI	Java	GC	≥2
FairPlay	SFDL (Java)	GC	2
FairPlayMP	SFDL (Java)	GC and Secret Sharing	≥2
TASTY	TASTYL (Python)	GC and HE	2

The VIFF, Sharemind, FairplayMP and TASTY frameworks can be easier to adapt since they are more accessible by using standard programming languages. Although, SCAPI is a preferable framework, since it is tailored for SMPC, it requires the users to be already knowledgeable on how the framework works. SCAPI also provides security with two instantiations, one for passive adversaries and the other for active. In contrast to the other frameworks, FairPlayMP requires an emulated TTP to receive the inputs from the parties, to perform the required computations and privately inform these parties about the outcome. In contrast to Fairplay, Tasty and VIFF, in which the adversarial tolerance is not mentioned, SCAPI, FairplayMP and Sharemind can withstand a potential corruption of the players. SCAPI tolerates both active and passive attackers, Sharemind can tolerate a passive adversary, which is capable to corrupt only one party and FairplayMP can withstand less than half of the players to be corrupted assuming that these players operate in a semi-honest manner.

[11] http://viff.dk/doc/overview.html

5. Conclusion

This deliverable provided a detailed description of the profiling approach used by the Cyber-Trust project to identify potential security risks along with the deployed components to attribute and profile malicious activities and threats at the device and network level. These components are Network architecture & assets repository [A16], Smart Device Agent (SDA), Profiling Service [A17], Trust Management Service [A05] and Intelligent Intrusion Response System (iIRS, A13). Further, this document gave an overview of the specific information that is being collected for the threat detection and mitigation, and trust assessment. Such information may characterize the device (i.e. device profile) or the network (i.e. network profile); examples include information about the integrity of a device's firmware and critical OS files, whether software patches have been installed, exposure to known vulnerabilities (in the enriched VDB), network behavioural patterns (e.g. IP addresses, VLANs, Ports and protocols), and services utilisation.

IoT device profiling is very useful in detecting potential attacks, especially new and unknown attacks, and advert malicious activities that may be missed by an Intrusion Detection System (IDS). However, such activity may introduce some security and privacy issues in which user's data can be misused or exploited. In fact, collected data contain personal and sensitive information (e.g. user identifier, username, hostnames, IP addresses, URLs, payloads, etc.) that may be attractive to cybercriminals. **To counter this, certain lightweight privacy-preserving techniques are already being used to perform pseudonymization, based on hash-functions constructions, and preserve user's privacy for data in transit.** Further, IoT devices come up with restricted resources including low power sources, small amounts of memory and limited processing power, which means minimizing the number of processes, and consequently, the size of the applications. These limitations hinder the execution of complex security tasks that generate massive computation and communication load. Consequently, appropriate solutions for these devices should maintain a balance between the high-security requirements and supporting infrastructures' hardware limitations. Therefore, the use of a fully homomorphic encryption (FHE) scheme at this stage could prevent due to its inherent complexity the proper exploration of the true capabilities of Cyber-Trust's solutions during the proof-of-concept pilot; the privacy benefits that could have been brought, were dealt with at the design stage of the project's architecture (adhering to the privacy-by-design principle). Thus, this document performed a thorough review of the current state-of-the-art in privacy-preserving methods and tools for IoT devices. The main findings of this report are summarized below.

- Cryptographic hash functions have been identified as the most suitable pseudonymization method for preserving user's privacy when collection and processing of personal data may be a concern. Thus, most prominent cryptographic hash functions have been studied and compared based on general and technical features that include the block size, the digest size and the word size, as well as the common attacks. SHA-2 was found to be the most efficient against attacks and provides a high security level. Thus, the SHA-256 hash function (widely used across industry) has been chosen to pseudonymise sensitive data that will be collected and processed by the cyber-Trust components.
- Lightweight encryption algorithms (LEA) (both symmetric and asymmetric) are also a suitable option to ensure anonymity and security in the IoT landscape. These algorithms are interesting for the Cyber-Trust project due to their ability to be executed at lower costs and with lower power consumption compared to the conventional cryptography. The AES algorithm is (as expected) the algorithm of choice when need to ensure the confidentiality of the platform's communications. However, LEA are the target of different kind of attacks (although some of them being rather theoretical), especially, Man-in-middle-attack, Saturation attack, Related Key attack, Differential attack, Timing attack and Module attack. Therefore, such solutions need to be further assessed for their use in privacy-preserving schemes in a future research project, where their employment might be well justified from the use cases (as advocated by the cryptographic community, there is no "one algorithm fitting all cases" solution).
- Other techniques that could be used to protect personal data shared between the Cyber-trust components are those in the area of Secure Multi-Party Computation (SMPC). SMPC is a generic

cryptographic technique that allows distributed parties to jointly compute an arbitrary function without revealing their private data. An overview and comparative evaluation of well-known approach that have been designed to carry out SMPC namely SCAPI, FairPlay, Sharemind and VIFF, FairPlayMP and TASTY. However, those techniques were not eventually adopted in Cyber-Trust's platform for the reasons exemplified above in the case of FHE.

6. References

[1] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, “A Survey on Security and Privacy Issues in Internet-of-Things,” *IEEE Internet Things J.*, 2017, doi: 10.1109/JIOT.2017.2694844.

[2] R. H. Weber, “Internet of things: Privacy issues revisited,” *Comput. Law Secur. Rev.*, 2015, doi: 10.1016/j.clsr.2015.07.002.

[3] M. Hildebrandt, “Defining profiling: A new type of knowledge?,” *Profiling Eur. Citiz. Cross-Disciplinary Perspect.*, pp. 17–45, 2008, doi: 10.1007/978-1-4020-6914-7_2.

[4] S. Gutwirth, R. Leenes, P. De Hert, and Y. Poullet, *European data protection: Coming of age*. 2013.

[5] K.-P. Wiedmann, H. Buxel, and G. Walsh, “Customer profiling in e-commerce: Methodological aspects and challenges,” *J. Database Mark. Cust. Strateg. Manag.*, 2002, doi: 10.1057/palgrave.jdm.3240073.

[6] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, “A Supervised Intrusion Detection System for Smart Home IoT Devices,” *IEEE Internet Things J.*, 2019, doi: 10.1109/JIOT.2019.2926365.

[7] M. Gajewski, J. Mongay Batalla, A. Levi, C. Togay, C. X. Mavromoustakis, and G. Mastorakis, “Two-tier anomaly detection based on traffic profiling of the home automation system,” *Comput. Networks*, 2019, doi: 10.1016/j.comnet.2019.04.013.

[8] H. U. & H. B. P. TARIQAHMAD SHERASIYA, “a Survey: Intrusion Detection System for Internet of Things,” *Int. J. Comput. Sci. Eng.* , 2016.

[9] H. Suo, J. Wan, C. Zou, and J. Liu, “Security in the internet of things: A review,” in *Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012*, 2012, doi: 10.1109/ICCSEE.2012.373.

[10] Z. Qiao, S. Liang, S. Davis, and H. Jiang, “Survey of attribute based encryption,” in *2014 IEEE/ACIS 15th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2014 - Proceedings*, 2014, doi: 10.1109/SNPD.2014.6888687.

[11] L. Sweeney, “k-anonymity: A model for protecting privacy,” *Int. J. Uncertainty, Fuzziness Knowlege-Based Syst.*, 2002, doi: 10.1142/S0218488502001648.

[12] J. J. V. Nayahi and V. Kavitha, “Privacy and utility preserving data clustering for data anonymization and distribution on Hadoop,” *Futur. Gener. Comput. Syst.*, 2017, doi: 10.1016/j.future.2016.10.022.

[13] ENISA, “Pseudonymisation techniques and best practices,” *ENISA website*. https://www.enisa.europa.eu/news/enisa-news/enisa-proposes-best-practices-and-techniques-for-pseudonymisation.

[14] ENISA, “Recommendations on shaping technology according to GDPR provisions - An overview on data pseudonymisation,” *ENISA website*, 2018. .

[15] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. 1996.

[16] M. Bellare, R. Canetti, and H. Krawczyk, “Keying hash functions for message authentication,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1996, doi: 10.1007/3-540-68697-5_1.

[17] R. C. H. Krawczyk,M. Bellare, “HMAC: Keyed-Hashing for Message Authentication.” [Online]. Available: https://tools.ietf.org/html/rfc2104.

[18] W. J. Buchanan, S. Li, and R. Asif, “Lightweight cryptography methods,” *J. Cyber Secur. Technol.*, 2017, doi: 10.1080/23742917.2017.1384917.

[19] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, “A lightweight block cipher for multiple

platforms," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, doi: 10.1007/978-3-642-35999-6_22.

[20] T. Suzaki and K. Minematsu, "Improving the generalized Feistel," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, doi: 10.1007/978-3-642-13858-4_2.

[21] D. Hong *et al.*, "HIGHT: A new block cipher suitable for low-resource device," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006.

[22] A. Bogdanov *et al.*, "PRESENT: An ultra-lightweight block cipher," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, doi: 10.1007/978-3-540-74735-2_31.

[23] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," *J. Ambient Intell. Humaniz. Comput.*, 2017, doi: 10.1007/s12652-017-0494-4.

[24] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Commun. ACM*, 1978, doi: 10.1145/359340.359342.

[25] S. Nakov, "Elliptic Curve Cryptography (ECC)," *Pract. Cryptogr. Dev.*, 2019.

[26] A. C. Yao, "PROTOCOLS FOR SECURE COMPUTATIONS.," in *Annual Symposium on Foundations of Computer Science - Proceedings*, 1982.

[27] P. R. Sousa, L. Antunes, and R. Martins, "The present and future of privacy-preserving computation in fog computing," in *Fog Computing in the Internet of Things: Intelligence at the Edge*, 2017.

[28] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen, "Asynchronous multiparty computation: Theory and implementation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, doi: 10.1007/978-3-642-00468-1_10.

[29] R. Cramer, I. Damgård, and Y. Ishai, "Share conversion, pseudorandom secret-sharing and applications to secure computation," in *Lecture Notes in Computer Science*, 2005, doi: 10.1007/978-3-540-30576-7_19.

[30] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, doi: 10.1007/978-3-540-88313-5-13.

[31] A. Ben-David, N. Nisan, and B. Pinkast, "FairplayMP - A system for secure multi-party computation," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2008, doi: 10.1145/1455770.1455804.

[32] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay - A secure two-party computation system," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[33] D. Beaver, S. Micali, and P. Rogaway, "Round complexity of secure protocols," 1990.

[34] R. Cramer and V. Shoup, "A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1998, doi: 10.1007/BFb0055717.

[35] M. Bellare and P. Rogaway, "Optimal asymmetric encryption," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1995, doi: 10.1007/bfb0053428.

[36] N. Fips, "197: Announcing the advanced encryption standard (AES)," *… Technol. Lab. Natl. Inst.*

Stand. …, 2001, doi: 10.1016/S1353-4858(10)70006-4.

[37] T. Elgamal, “A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms,” *IEEE Trans. Inf. Theory*, 1985, doi: 10.1109/TIT.1985.1057074.

[38] W. Henecka, S. Kögl, A. R. Sadeghi, T. Schneider, and I. Wehrenberg, “TASTY: Tool for automating secure two-party computations,” in *Proceedings of the ACM Conference on Computer and Communications Security*, 2010, doi: 10.1145/1866307.1866358.