MontaVista VPN Client (MVC)
Common Criteria Security Target

Version 0.042
FINAL (provisional)
2021-11-03

Prepared By:
MontaVista Software, LLC
5201 Great America Parkway, Suite 432
Santa Clara, CA 95054
USA

Contents

Figures

Tables

Change History

Version	Change Description	Date
V003	Pre-PRELIMINARY	31 July 2020
V004	PRELIMINARY	6 August 2020
V006	PRELIMINARY	10 August 2020
V009	PRELIMINARY	31 August 2020
V014	PRELIMINARY	21 September 2020
V015	DRAFT (first version meeting criteria for DRAFT)	24 September 2020
V016	DRAFT Changed Figure 1; added Change History; formatting	28 September 2020
V017	DRAFT edits and initial modifications from review	7 October 2020
V018	DRAFT changes due to 1st atsec review	18 October 2020
V018a	DRAFT Incorporated RS changes	19 October 2020
V019	DRAFT changed SFR numbering and order	20 October 2020
V020	DRAFT responses to incremental atsec review; rationales	31 October 2020
V021	DRAFT edits to section 1; misc edits	3 November 2020
V022	DRAFT delete arch design summary; add to TOE overview; misc edits; glossaries	11 November 2020
V023	DRAFT edits to SFRs and EXT component defs; misc edits	29 November 2020
V024	FINAL DRAFT	1 December 2020
V025	FINAL DRAFT some additions	7 December 2020
V026	FINAL DRAFT atsec comments (multiple dated versions) w/markup	29 January 2021
V027	FINAL (provisional) completed comments; PLT crypto SFRs; w/markup	8 February 2021
V028	FINAL (provisional)	17 February 2021
V029	FINAL (provisional): Update based on evaluator comments	25 February 2021
V030	FINAL (provisional): addressed some issues from evaluator comments	28 February 2021
V031	FINAL (provisional): changes from evaluator comments	7 March 2021
V032	FINAL (provisional): more changes from evaluator comments	18 March 2021
V033	FINAL (provisional): Clarifications and updates made based on the QA review comments	5 April 2021
V034	FINAL (provisional): Minor updates and clarifications	15 April 2021
V035	FINAL (provisional): Minor updates and clarifications	26 April 2021
V036	FINAL (provisional): added guidance doc name and FPT_TST details	14 July 2021
V037	FINAL (provisional): minor fixes to the FPT_TEE and FPT_TST	25 August 2021
V038	FINAL (provisional): physical scope, typographical/grammatical edits	31 August 2021
V039	FINAL (provisional): CGX 2.6 source repo and distribution id strings	13 September 2021
V040	FINAL (provisional): Update to kernel key erasure	26 September 2021
V041	FINAL (provisional): Update to the delivery	29 September 2021
V042	FINAL (provisional): Added clarification to testing in section 1.4.2	3 November 2021

1 Security Target (ST) Introduction

The structure of this document is defined by CC v3.1R5 Part 1 Annex A, Section A.2 "Mandatory contents of an ST" [1].

1.1 Security Target Reference

ST Title:	MontaVista VPN Client (MVC) Common Criteria Security Target
ST Version Number:	Version 0.042 (FINAL (provisional))
ST Author(s):	Rance DeLong and Riku Salminen and Markus Veranen
ST Publication Date:	2021-11-03
Keywords:	IPsec, VPN, client

1.2 TOE Reference

TOE Developer	MontaVista Software, LLC 5201 Great America Parkway, Suite 432 Santa Clara, CA 95054 USA
TOE Name:	MontaVista VPN Client (MVC)
TOE Software Version:	MVC 1.0

1.3 TOE Overview

The TOE is part of the MontaVista VPN Client (MVC). The MVC is a software product that provides an VPN IPsec connection between the client (TOE) and a server.

The MVC is employed by an end-user as a client on an operating environment to establish a mutually-authenticated trusted channel with a server on a remote system running a compatible implementation of the standard protocols implemented by TOE.

The summary of the security functionality of the TOE is:

- The TOE provides data confidentiality and integrity, and protects against replay and disclosure and manipulation attacks against data transmitted within the trusted channel.
- TOE utilizes the Internet Key Exchange version 2 (IKEv2) protocol and the IPsec Encapsulating Security Payload (ESP) protocol. These protocols provide strong mutual authentication of the TOE's VPN peer and negotiate keys to establish one or more trusted channels for confidentiality- and integrity-protected communication between endpoints over both IPv4 and IPv6 network protocol implementations provided by the operating environment.
- The TOE generates audit records for security-relevant events and forwards them to audit logging and storage facilities provided by the operating environment.
- Self-tests to assure the correct operation of security functions.

The MVC is intended to be used with the MontaVista Linux Carrier Grade eXpress (CGX) operating system using the ARM processor architecture.

The MVC consists of a collection of software modules that include components from the open-source software project strongSwan and Linux kernel networking components. The TOE is provided as a binary executables along with the whole MVC source code, including all of the non-TOE components and tools necessary to build the executable images. However, only the binary executables and not the source code version is considered the TOE. The scope of the TOE is shown in Figure 1 below.

1.4 TOE Description

The MVC is used to establish a cryptographically secure data communication channel between a local user and a remote trusted user or to establish a trusted network over a potentially unsafe network. We will refer to such data, as well as configuration data that is critical to the security of the VPN client, as sensitive or "red" data. An unsafe network we will refer to as a "public network" or "black network."

The client is physically realized in the MVC as a software product (identified as an instance of "Variant 2" in [11]). The core functionality is to transmit red data with confidentiality, integrity and authenticity achieved by establishing a suitably configured VPN tunnel between the VPN client and a compatible VPN gateway. Supporting functions needed to protect the VPN client and to configure and establish the secure channel, perform encryption/decryption and signing of data, key and certificate management, key storage, etc. are provided by the platform running the MVC product. The MVC utilizes features of the non-TOE hardware/firmware/software platform

provided by its environment, including cryptographic operations, key management and key storage.

Figure 1 illustrates the extended architecture of the MVC, including the TOE components, and the platform (PLT) components. It distinguishes among the TOE and the PLT by color. Of note among the non-TOE components is the Configuration Agent component of the platform. The TOE and other PLT components provide configuration interfaces but the actual Configuration Agent that uses those interfaces is to be provided by the integrator, who will determine the manner in which the MVC fits into the operational workflow and management of the larger system in which MVC is employed. The figure also depicts whether an TOE or PLT component runs in the CGX kernel or in user space.

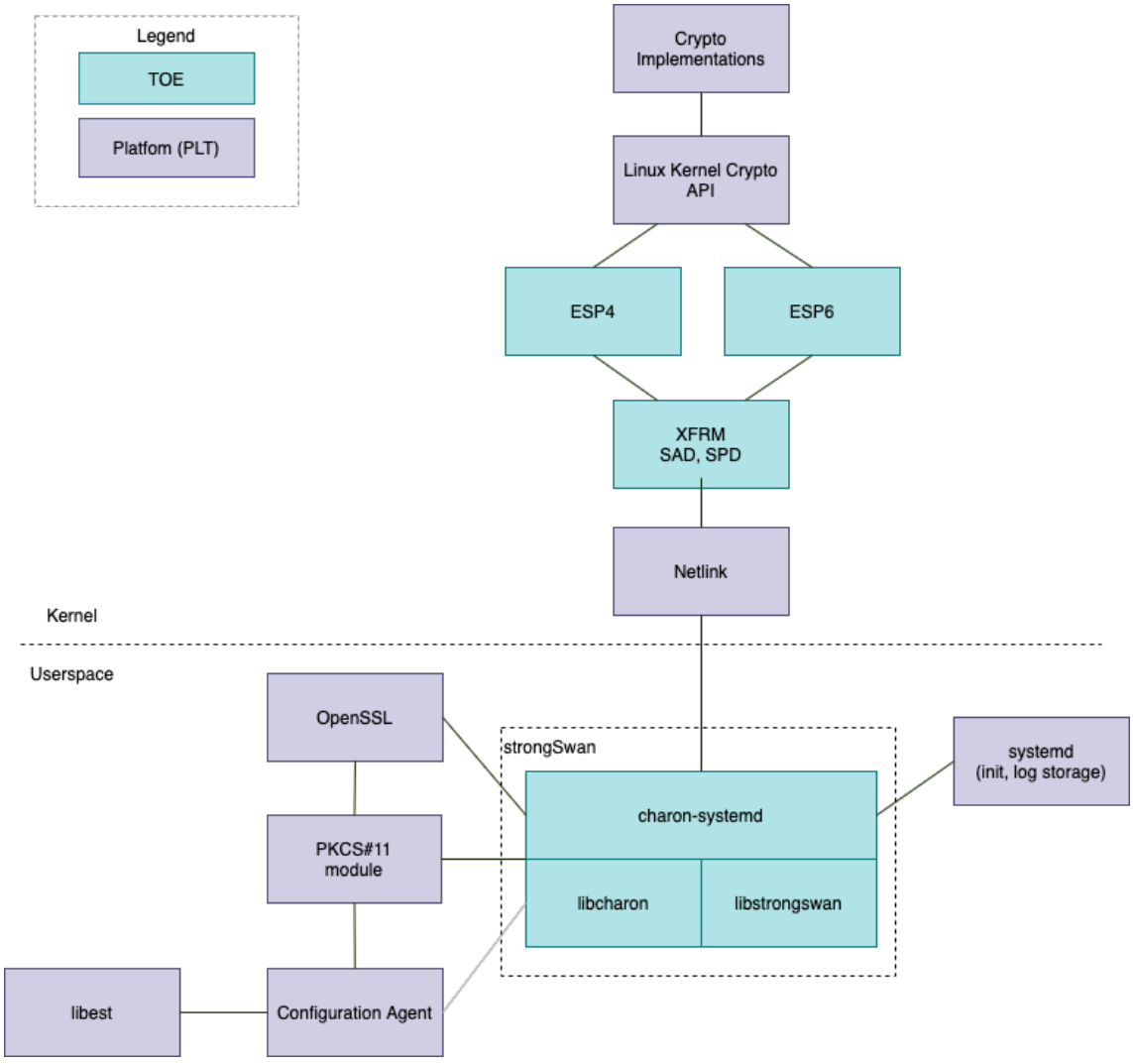


Figure 1: TOE and platform components of extended MVC architecture

1.4.1 TOE software components

Encapsulating Security Payload (ESP), Security Policy Database (SDP) and Security Association Database (SAD) are TOE functions implemented by the following Linux kernel components:

- ESP4 (MVC) implements ESP for Internet Protocol version 4 (IPv4).
- ESP6 (MVC) implements ESP for Internet Protocol version 6 (IPv6).
- XFRM (MVC) implements the Security Policy Database (SPD) and the Security Association Database (SAD).

Internet Key Exchange version 2 (IKEv2) is implemented by the following MVC strongSwan charon keying daemon components:

- charon-systemd implements charon keying daemon systemd integration.
- libcharon implements most of the charon IKEv2 daemon functionality.
- libstrongswan is the foundation library of the charon IKEv2 keying daemon.

1.4.2 Platform software components required for operation of the MVC

The MVC relies on platform-provided functionality for identification and authentication, file system, access control, log storage, networking support (IP/UDP), cryptographic algorithm implementations, secure key storage and other operating system services.

The platform may provide discrete user identities and provide an identification and authentication procedure to bind a specific user identity to an instance of the execution of the MVC or it may simply preserve and associate an identity established by an external administrative procedure, such as assignment of a device having an implicit associated identity to a person. MVC connections to environment resources such as files or network ports, or to subjects such as a user or administrator subject is done through associations by the platform between such resources or subjects and the interfaces of the MVC.

Cryptographic algorithm implementations for the kernel space ESP implementation are provided by the platform and accessed via the Linux Kernel Crypto API. For the kernel part the MVC relies on the Linux Kernel Crypto API. During startup, the TOE runs a set of tests on the algorithms that it relies upon.

Cryptographic algorithm implementations for the user space charon-systemd IKEv2 implementation are provided by the OpenSSL library (platform) and a PKCS#11 module. By default, the MVC platform uses SoftHSM, a software only PKCS#11 implementation. SoftHSM is configured to use cryptographic algorithm imlementations provided by the OpenSSL library.

The charon-systemd daemon controls the Security Policy Database (SPD), which specifies what security services are to be applied to IP packets and how, and the Security Association Database (SAD), which determines whether a packet is subject to IPsec processing and the processing details. These databases are maintained in kernel space through the Netlink Socket API. The same API is used for communication from the kernel space IPsec stack to the charon-systemd daemon.

The platform provides the systemd init daemon. Systemd starts charon-systemd and collects and stores log information generated by charon-systemd.

The Configuration Agent is a platform component provided by the integrator. It provides the user interface for configuration and interfaces to other parts of the user's system. The Configuration Agent uses libest. Libest implements RFC 7030, Enrollment over Secure Transport (EST), and uses OpenSSL.

During the evaluation, the TOE was tested against an independent reference implementation of IPsec/IKE. That reference implementation is part of SUSE Linux Enterprise Server 15 SP2 which was CC certified by BSI on 2021-07-08. This would also allow the evaluator to verify not only the TOE's implementation of IPsec and IKE protocols, but also the underlying cryptographic primitives that are provided by the operational environment.

1.4.3 Operating environment for execution of the MVC

MontaVista Linux Carrier Grade eXpress (CGX) is MontaVista's main operating system product that delivers Carrier Grade reliability, security, and serviceability in a highly configurable, flexible package with consistent high quality. CGX is compliant with the Carrier Grade Linux (CGL) 5.0 standard which in turn complies with the Linux Standard Base (LSB) 3.0. A goal of the CGL working group is to promote migration from proprietary hardware platforms to COTS hardware by assuring adequate support for COTS platforms.
Considering the diversity of hardware platforms, the CGL working group defines generic platform requirements and then provides an "Industry Platforms" section with guidelines for specific architectures.

CGX meets key criteria set by the CGL working group, including IPv6, cipher, IPsec, IKE daemon support, role-based access control, auditing, file access tracing, and tamper resistant storage for security-relevant data such as keys and certificates.

The primary MVC configuration and management functions are delegated to a Configuration Agent that is run within the IT environment referred to here as the platform, which is provided by the integrator and uses the configuration interfaces provided by the MVC.

1.4.4 Hardware/firmware platform for execution of the TOE/PLT software components

The TOE requires a hardware/firmware platform that is compatible with the CGX operating system, has at least one network interface and provides persistent storage for software components, configuration and log data. The TOE does not interface to the hardware directly, but only through interfaces provided by the platform's operating environment. The evaluation platform hardware is an AGIB A101 board, secure SoC environment (ARM ISA).

1.4.5 MVC Security features

The following security features are provided by the MVC:

1. Cryptographic protocols capable of successful runs despite passive eavesdropping or active manipulations of coordination and data communication messages in transit by an adversary with powers such as those defined by the Dolev-Yao symbolic model
2. Cryptographic key negotiation and handling
3. Establish a mutually authenticated trusted channel "VPN tunnel" over a public "black" network
4. Confidentiality and integrity of red data-in-transit through VPN tunnel
5. Generation of audit records for security-relevant events occurring within the MVC

6. Self-tests to assure the correct operation of supporting functions.

MVC implements security functionality that integrates with the CGX kernel to provide IPSec ESP packet path functionality, maintaining Security Association & Security Policy databases (SAD and SPD) and enforcing IPsec ESP protocol transformation of network packets. In CGX user space, MVC implements security functionality for managing IPsec configuration, running IPsec IKEv2 keying protocol and controlling ESP transformation in the kernel by applying SA and SP information.

1.4.5.1 Cryptographic protocols

The MVC implements ESP protocol to provide confidentiality, data origin authentication, connectionless integrity, and anti-replay service to connections between local and remote red networks for both ipv4 and ipv6. The MVC performs encrypting and signing by using algorithms which fulfill the requirements for VS-NfD classified information [11], to mitigate against eavesdropping and manipulation of messages while in transit over the black network.

For a detailed description of the cryptographic algorithms and keys used, please see section 7.3.1 and especially Table 14 – Cryptographic algorithms and Table 15 – Cryptographic keys.

1.4.5.2 Cryptographic key negotiation and handling

IKEv2 protocol is implemented by the MVC for IPsec keying functionality. At phase 1 the MVC performs mutual authentication of the peers using X.509 certificates, negotiates cryptographic parameters, and creates session keys for the rest of IKEv2 communication (Phase 2).

1.4.5.3 Establish mutually authenticated trusted channel

IPsec ESP functionality of the MVC implements IPsec tunnel mode, which encapsulates the whole IP packet by encrypting and authenticating the original IP packet. Encryption and authentication is performed by utilizing keys and algorithm selections from an Security Association (SA) entry in SA Database. The SA entry is created for the connection by IKEv2 functionality in Phase 2 negotiation.

1.4.5.4 Confidentiality and Integrity of red data-in-transit

The red data in-transit is protected by encapsulating it into IPsec VPN ESP tunnel, negotiated between local and remote peer as described above. For network packets, IPsec Security Policy Database is searched to determine whether a packet is subject to IPsec transformation.

1.4.5.5 Generation of audit records

StrongSwan generates audit records of IPsec IKEv2 security-relevant events via the systemd logging interface provided by the platform. The log entries are stored into a file and accessible locally to the security administrator.

1.4.5.6 Self-test of the cryptographic functions

During start-up, the TOE will perform test of the strongSwan and test of the cryptographic functions that are provided by the platform (both the kernel and the user space part). The TOE will also test the integrity of the strongSwan configuration. If these tests are failing the TOE will not be operational and will enter a secure state. In that case the TOE will generate an audit event.

1.4.6 MVC life cycle

The MV is provided as binary images along with a minimal supporting software platform in form of a the CGX 2.6 deliverable reflecting the version and evaluated configuration of the TOE.

In addition, the MVC is provided as source code to the integrator as network download containing a Docker image, which in turn contains all software components of the MVC test configuration, including the PLT, CGX, third party components needed to complete the MVC architecture, and a simple generic placeholder component to take the place of the Configuration Agent that is to be replaced by the integrator supplied version. The integrator will combine the MVC with its own software components and hardware to create its own product that includes the MVC and will provide delivery, acceptance procedures, and user guidance for the product.

MVC is provided along with guidance for the integrator in the document CGX 2.6 Getting Started Guide TI 66AK2H (AGIB) that covers the use of the MVC delivery and the establishment of the necessary operational environment for tasks that the integrator must perform with the MVC. The integrator guidance includes sections reflecting the objectives for the operational environment that are intended for the integrator to pass through by inclusion in the User Guidance that the integrator provides to the end users of its product. The integrator will need to include product-specific installation and administration information in the User Guidance it provides.

1.4.7 Physical Scope

The TOE is delivered as binary images along with a minimal supporting software platform in form of a the CGX 2.6 deliverable. The TOE is delivered by download from the MontaVista "Zone" at https://support.mvista.com. The binary images are considered to be the TOE, along with the documentation as described below. Additionally, the source code is available for download and build using the provided build tools and procedures.

The MVC delivery consists of the following items:

- A Docker image containing images comprising a minimal executable CGX system, binary images of the MVC components identified in Section 1.4 as built from the source repository, and the environment and tools required for re-building the MVC and its platform components from source code;
- Pre-built binary image(s) of CGX including MVC, suitable for execution on the target hardware, as the CGX 2.6 deliverable with buildtag "agib-4.19-2.6_210823084105";
- Documentation in digital form within the physical distribution media containing the instructions for installing and executing the contents of the distribution area, guidance for integration of integrator-supplied components, guidance for preparation of a suitable execution environment for the MVC based on the environment objectives for the MVC, and guidance that should be passed through to the end user as part of integrator-provided user guidance; and
- A PDF file of the CGX 2.6 Getting Started Guide and a description of how to access the MontaVista Zone and the balance of the documentation in digital form.

In addition, a collection of Git repositories is made available, containing the source of the MVC and of platform components required to support the MVC. The Git branch name of the source

repository is “thud-210823084105”. From this source the binary images of the MVC and PLT can be built. The source code is available to be downloaded by the integrators for development and integration activities. The source repositories may be either mounted or cloned to the Docker container. Build may be run inside the container and resulting binary image(s) copied from the Docker container to the host system.

1.4.8 Logical Scope

The logical scope is described in section 1.4.5 and the subsections 1.4.5.1 to 1.4.5.7.

2 Conformance claims

2.1 Common Criteria Conformance Claims

This ST is CC Part 2 extended and CC Part 3 conformant.

This ST claims conformance to CC version 3.1 Revision 5, April 2017.

This ST claims conformance to the EAL4 package of security assurance requirements augmented with ALC_FLR.2 and AVA_VAN.4.

This ST does not claim conformance to any protection profile.

2.2 Conformance rationale

The MVC is to be applied in environments wherein at most VS NfD “restricted” information is handled and to be processed by the MVC. The choice of EAL4 conformance and its augmentation is based on precedent established by BSI in Germany. Specifically, these assurance requirements are based on the VS-Anforderungsprofil VPN-Client (VPNC), 19.09.2018 [11].

3 Security problem definition

3.1 Preliminaries

Terms and abbreviations used in this security problem definition and elsewhere in the security target are defined in Section 8.

The assets of concern with respect to the MVC are:

- User- or organization-sensitive data (*red data*)
- TSF data such as configuration data or internal data (also considered red data)

Data assets occur as data-at-rest (DAR), data-in-use (DIU), or data-in-transit (DIT).
Note that user authentication data as well as long-term keys and certificates are assets that are managed by and stored within the platform, not under direct control of the TOE.

3.2 Threats

The following subsections define the security threats for the MVC, characterized by a threat agent ("adversary"), an asset, and an adverse effect on that asset caused by the threat agent.

The threats listed in Table 1 are to be countered by the MVC operating on its platform within the operating environment.

Table 1 – Threats

Threat	Description
T.NETWORK_ATTACK	The actions of an adversary on the black network (including message inspection, disassembly, replay, deletion, modification, and creation potentially across multiple sessions) enable the adversary to defeat the objectives of the security protocols implemented by the TOE resulting in violation of a security policy, such as the confidentiality or integrity of red DIT.

3.3 Organizational security policies

The OSPs listed in Table 2 are intended to represent generic policies expected to be common to organizations using the MVC.

Table 2 – Organizational Security Policies

OSP	Description
P.LOGGING	Information regarding the occurrence of security-relevant events within the TSF shall be logged for later forensic examination, including the identity of associated individual user or subject, along with other relevant particulars.
P.RED_DATA_PROT	The TOE shall ensure that red data is protected when it is under control of the TOE and that it is only transmitted over the designated communication channel to or from a peer over a black network.
P.SELF_TEST	The TOE must verify the integrity of the configuration and verify that the cryptographic operations are performed correctly before any outside connections is established. If the test fails the TOE must come to a halt.

3.4 Assumptions on the MVC operational environment

This section describes the assumptions that are made on the operational environment in which the MVC runs in order to be able to provide its security functionality. It includes information about the physical, personnel, and IT platform aspects of the environment.

The operational environment must be managed in accordance with the provided guidance documentation. Table 3 enumerates specific conditions that are assumed to exist in the environment and in the IT system upon which the TOE operates (those prefixed with "IT_").

Table 3 – Assumptions on the Operational Environment

Assumption	Description
A.PHYSICAL_SECURITY	The non-IT environment provides the TOE and the platform upon which it operates with appropriate physical security to prevent tampering commensurate with the value of the IT assets protected by the TOE.
A.PERSONNEL	Personnel that use or administer the TOE or the platform are assumed to be trusted, trained and follow all applicable guidance documentation.
A.IT_ADMIN	The operational environment of the TOE provides procedures and tools for the secure configuration and administration of the TOE.
A.IT_STORAGE	The IT environment provides protected persistent storage for programs and data of the TOE
A.IT_ACCESS_CONTROL	The IT environment for the operation of the TOE provides appropriate and adequate access control for assets upon which the TOE depends, including: TOE executable files, configuration files, ports, or other interfaces.
A.IT_CRYPTO_EXP	The cryptographic primitives, RNG and memory management for kernel key erasure required by the TOE are provided by the underlying platform.
A.IT_CRYPTO_GEN	The IT environment provides mechanisms for the storage, distribution and management of cryptographic private keys and certificates.
A.IT_INITIALIZATION	The IT environment has hardware and software features to ensure correct establishment of initial secure state.
A.IT_TIME	The IT environment has hardware and software features to provide the current time.
A.IT_MEDIATION	All access to data assets (including red data and external network connections) that exist in the IT environment is mediated by and subject to the controls provided by the platform upon which the TOE executes.

4 Security objectives

4.1 Security Objectives for the TOE

The security objectives for the TOE are listed in Table 4.

Table 4 – Security Objectives for the TOE	
TOE Objective	Description
O.COMMUNICATION	TOE establishes a mutually authenticated trusted communication channel with its peer. The trusted channel provides confidentiality and integrity of red data-in-transit over the black network. The cryptographic/communication protocols implemented by the TOE ensure that red data does not reach the black network as cleartext, and thwart network attacks such as replay and other message manipulation attacks
O.AUDIT	Select security-relevant events in the TOE are logged for later inspection
O.SELF_TEST	The TOE must verify the integrity of the configuration and verify that the cryptographic operations are performed correctly before any outside connections is established. If the test fails the TOE must come to a halt.
O.INFO_FLOW	TOE processing and communication prevents modification of red data by, or disclosure of red data to, an untrusted subject by forcing red data to flow only to trusted subjects through trusted channels.

4.2 Security Objectives for the Operational Environment

The security objectives for the operational environment are listed in Table 5.

Table 5 – Security Objectives for the Operational Environment	
Objective	Description
OE.PHYSICAL_SECURITY	Physical access to the environment and to the platform for the execution of the TOE is restricted to authorized individuals
OE.PERSONNEL	Personnel that access, operate, or maintain the TOE or the platform are trained to follow all applicable guidance documentation and are deemed trustworthy at a level commensurate with the value of the data assets (may be facilitated by background investigations and other vetting)
OE.PT_ADMIN	The operational environment of the TOE provides procedures and tools for the secure configuration and administration of the TOE.
OE.PT_FILE_STORAGE	The platform provides persistent and access-controlled storage of data-at-rest including TOE configuration, cryptographic keys and certificates, and audit logs
OE.PT_ACCESS_CONTROL	The platform provides adequate enforcement of an access control policy at appropriate granularity for assets processed by or upon which the TOE depends, including: TOE executables, configuration files, ports, or other interfaces
OE.PT_CRYPTO_EXP	The platform provides cryptographic primitives, RNG and memory management for kernel key erasure required by the TOE.
OE.PT_CRYPTO_GEN	The platform provides mechanisms for the storage, distribution and management of cryptographic private keys and certificates.
OE.PT_INITIALIZATION	The platform provides a secure boot procedure that ensures that the hardware, firmware and software components of the platform and the TOE are using uncorrupted instances.
OE.PT_TIME	The platform provides the TOE with a source of reliable and accurate time.
OE.PT_SEP	The platform provides separation of the TOE from other applications and separation of TSF data from other data, preventing infiltration to and exfiltration from the TOE's processes and data areas by other processes.
OE.PT_RVM	The platform provides mediation of data flows to and from the TOE to achieve non-bypass-ability of the security functions provided by the TOE.

4.3 Rationale for the Security Objectives

Table 6 presents the mapping of the threats, organizational security policies, and environmental assumptions to the security Objectives for the TOE (O) and Objectives for the Environment (OE), including the specific objectives for the platform (OE.PT). The appearance of and 'X' indicates that a threat, policy or assumption is addressed in some measure by the objective. The manner in which the objectives contribute to the elimination or mitigation of each threat, or to the enforcement of each policy are discussed subsequently.

Table 6 – Security Problem to Security Objectives Mapping

	O.COMMUNICATION	O.AUDIT	O.SELF_TEST	O.INFO_FLOW	OE.PHYSICAL_SECURITY	OE.PERSONNEL	OE.PT_ADMIN	OE.PT_FILE_STORAGE	OE.PT_ACCESS_CONTROL	OE.PT_CRYPTO_EXP	OE.PT_CRYPTO_GEN	OE.PT_INITIALIZATION	OE.PT_TIME	OE.PT_SEP	OE.PT_RVM
T.NETWORK_ATTACK	X														
P.LOGGING		X													
P.SELF_TEST			X												
P.RED_DATA_PROT	X			X											
A.PHYSICAL_SECURITY					X										
A.PERSONNEL						X									
A.IT_ADMIN							X								
A.IT_STORAGE								X							
A.IT_ACCESS_CONTROL									X						
A.IT_CRYPTO_EXP										X					
A.IT_CRYPTO_GEN											X				
A.IT_INITIALIZATION												X			
A.IT_TIME													X		
A.IT_MEDIATION														X	X

The application to each threat and policy of the Objectives for the TOE, and for each threat, policy, and assumption of the Objectives for the Environment are described in the following:

- T.NETWORK_ATTACK – Network attacks are addressed by O.COMMUNICATION. O.COMMUNICATION requires establishment of a mutually authenticated trusted channel that has the properties: confidentiality and integrity of red data and resistance to network message and protocol manipulation attacks. This threat is deemed to be eliminated by the cited security objectives.
- P.SELF_TEST – The organizational security policy of self test is mitigated by the TOE objective O.SELF_TEST, which is intended to detect loss of integrity in the configuration of the TOE and to verify that the cryptographic operations are performed correctly before any outside connections is established. The tests are performed dusing startip and if the test fails the TOE must come to a halt.

- P.LOGGING – The organizational security policy that there be logging of security-relevant events for forensic purposes is served by the O.AUDIT objective.
- P.RED_DATA_PROT – The protection of the confidentiality and integrity of red data is the overarching organizational security policy, and it is enforced within the TOE and during its transmission over the VPN. The objective O.COMMUNICATION achieves protection of red data over the black network through mutual authentication and encrypted communication. The objective O.INFO_FLOW concerns the protection and confinement of red data coming into and used within the TOE, assuring that red data does not reach the black network in cleartext form and is not improperly exported by the TOE. This policy is complemented by the assumption A.IT_MEDIATION that the platform will provide protection of red data flowing into and out of the TOE from other domains in the platform.
- A.PHYSICAL_SECURITY – The assumption of physical security is guaranteed by the operational environment objective OE.PHYSICAL_SECURITY.
- A.PERSONNEL – The assumption of users that are trusted, trained and follow applicable guidance is guaranteed by the operational environment objective OE.PERSONNEL that ensures that only trusted and trained users are granted access to the TOE and are vetted to an appropriate level of trustworthiness.
- A.IT_ADMIN – The assumption of administrative procedures and tools for secure configuration and administration of the TOE by the security objective OE.PT_ADMIN.
- A.IT_STORAGE – The assumption that the IT environment provides protected persistent storage for programs and data associated with the TOE and PLT is met by the platform objective OE.PT_FILE_STORAGE.
- A.IT_ACCESS_CONTROL – The assumption that the IT environment provides access control for TOE and PLT executable files, configuration files, ports, etc. is guaranteed by the platform objective OE.PT_ACCESS_CONTROL.
- A.IT_CRYPTO_EXP – The assumption that the platform provides the cryptographic primitives, RNG and memory management for key erasure required by the TOE is guaranteed by the platform objective OE.PT_CRYPO_EXP.
- A.IT_CRYPTO_GEN – The assumption that the platform provides mechanisms for the storage, distribution and management of cryptographic private keys and certificates is guaranteed by the platform objective OE.PT_CRYPTO_GEN.
- A.IT_INITIALIZATION –The assumption that the IT environment provides secure initialization to guarantee that key elements of the TOE and PLT are uncorrupted is met by the platform objective OE.PT_INITIALIZATION.
- A.IT_TIME – The assumption that the IT environment provides an accurate and trustworthy source of the current time is met by the platform objective OE.PT_TIME.
- A.IT_MEDIATION – The assumption that the platform mediates all access to and transfer of data assets is addressed by OE.PT_SEP and OE.PT_RVM which guarantee that the mediation mechanisms are tamperproof and non-bypassable.

Table 7 indicates whether Objectives for the Environment are met by the platform or by other non-IT environment conditions. ‘P’ indicates that the Platform (IT environment) meets an environment objective; ‘E’ indicates that a non-IT environment condition meets an environment objective. All of these Objectives for the Environment are achieved through the application of the provided guidance documentation.

Table 7 – Allocation of OEs

	PLT (IT environment)	ENV (non-IT environment)
OE.PHYSICAL_SECURITY		E
OE.PERSONNEL		E
OE.PT_ADMIN	P	
OE.PT_FILE_STORAGE	P	
OE.PT_ACCESS_CONTROL	P	
OE.PT_CRYPTO_EXP	P	
OE.PT_CRYPTO_GEN	P	
OE.PT_INITIALIZATION	P	
OE.PT_TIME	P	
OE.PT_SEP	P	
OE.PT_RVM	P	

5 Extended components definition

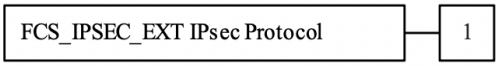
5.1 Security functional requirement extended components

This ST incorporates extended SFR definitions for FCS_IPSEC_EXT and FCS_RBG_EXT from the collaborative Protection Profile for Network Devices (cPPND) [6]. For completeness of the ST, the extended SFR definitions are copied below. Modifications made to the cPPND component definitions appear as refinements when these components are invoked in the security requirements for the TOE in Section 6.1.

5.1.1 FCS_IPSEC_EXT.1 IPsec Protocol

Family Behaviour
Components in this family address the requirements for protecting communications using IPsec.

Component levelling



FCS_IPSEC_EXT.1 IPsec requires that IPsec be implemented as specified.

Management: FCS_IPSEC_EXT.1
The following actions could be considered for the management functions in FMT:
a) Maintenance of SA lifetime configuration

Audit: FCS_IPSEC_EXT.1
The following actions should be considered for audit if FAU_GEN Security audit data generation is included in the PP/ST:
a) Decisions to DISCARD, BYPASS, PROTECT network packets processed by the TOE.
b) Failure to establish an IPsec SA
c) IPsec SA establishment
d) IPsec SA termination
e) Negotiation “down” from an IKEv2 to IKEv1 exchange.

FCS_IPSEC_EXT.1 Internet Protocol Security (IPsec) Communications
Hierarchical to: No other components
Dependencies: FCS_CKM.1Cryptographic KeyGeneration
FCS_CKM.2Cryptographic KeyEstablishment
FCS_COP.1/DataEncryption Cryptographic operation (AES Data encryption/decryption)
FCS_COP.1/SigGen Cryptographic operation (Signature Generation and Verification)
FCS_COP.1/HashCryptographic operation (HashAlgorithm)
FCS_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)
FCS_RBG_EXT.1 Random Bit Generation

FCS_IPSEC_EXT.1.1 The TSF shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2 The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched and discards it.

FCS_IPSEC_EXT.1.3 The TSF shall implement [selection: *tunnel mode, transport mode*].

FCS_IPSEC_EXT.1.4 The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [selection: *AES-CBC-128 (RFC 3602), AES-CBC-192 (RFC 3602), AES-CBC-256 (RFC 3602), AES-GCM-128 (RFC 4106), AES-GCM-192 (RFC 4106), AES-GCM-256 (RFC 4106),*] together with a Secure Hash Algorithm (SHA)-based HMAC [selection: *HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, no HMAC algorithm*].

FCS_IPSEC_EXT.1.5 The TSF shall implement the protocol: [selection:

• IKEv1, using Main Mode for Phase 1 exchanges, as defined in RFCs 2407, 2408, 2409, RFC 4109, [selection: no other RFCs for extended sequence numbers, RFC 4304 for extended sequence numbers], and [selection: no other RFCs for hash functions, RFC 4868 for hash functions];

• IKEv2 as defined in RFCs 5996 [selection: with no support for NAT traversal, with mandatory support for NAT traversal as specified in RFC 5996, section 2.23)], and [selection: no other RFCs for hash functions, RFC 4868 for hash functions]].

FCS_IPSEC_EXT.1.6 The TSF shall ensure the encrypted payload in the [selection: *IKEv1, IKEv2*] protocol uses the cryptographic algorithms [selection: *AES-CBC-128, AES_CBC-192 AES-CBC-256 (specified in RFC 3602), AES-GCM-128, AES-GCM-192, AES-GCM-256 (specified in RFC 5282)*].

FCS_IPSEC_EXT.1.7 The TSF shall ensure that [selection:

• IKEv1 Phase 1 SA lifetimes can be configured by a Security Administrator based on [selection:

o number of bytes;

o length of time, where the time values can be configured within [assignment:

integer range including 24] hours;];

• IKEv2 SA lifetimes can be configured by a Security Administrator based on [selection:

o number of bytes;

o length of time, where the time values can be configured within

[assignment: integer range including 24] hours

]].

FCS_IPSEC_EXT.1.8 The TSF shall ensure that [selection:

• IKEv1 Phase 2 SA lifetimes can be configured by a Security Administrator based on [selection:

o number of bytes;

o length of time, where the time values can be configured within [assignment:

integer range including 8] hours;

];

• IKEv2 Child SA lifetimes can be configured by a Security Administrator based on [selection:

o number of bytes;

o length of time, where the time values can be configured within [assignment:

integer range including 8] hours;

]].

FCS_IPSEC_EXT.1.9 The TSF shall generate the secret value x used in the IKE Diffie- Hellman key exchange (“x” in gx mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [assignment: *(one or more) number(s) of bits that is at least twice the security strength of the negotiated Diffie-Hellman group*] bits.

FCS_IPSEC_EXT.1.10 The TSF shall generate nonces used in [selection: *IKEv1, IKEv2*] exchanges of length [selection:

• *according to the security strength associated with the negotiated Diffie-Hellman group;*

• *at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash*

].

FCS_IPSEC_EXT.1.11 The TSF shall ensure that IKE protocols implement DH Group(s) [selection:

• *[selection: 14 (2048-bit MODP), 15 (3072-bit MODP), 16 (4096-bit MODP), 17 (6144-bit MODP), 18 (8192-bit MODP)] according to RFC 3526,*

• *[selection: 19 (256-bit Random ECP), 20 (384-bit Random ECP), 21 (521-bit Random ECP), 24 (2048-bit MODP with 256-bit POS)] according to RFC 5114.*

].

FCS_IPSEC_EXT.1.12 The TSF shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: *IKEv1 Phase 1, IKEv2 IKE_SA*] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: *IKEv1 Phase 2, IKEv2 CHILD_SA*] connection.

FCS_IPSEC_EXT.1.13 The TSF shall ensure that all IKE protocols perform peer authentication using [selection: *RSA, ECDSA*] that use X.509v3 certificates that conform to RFC 4945 and [selection: *Pre-shared Keys, no other method*].

FCS_IPSEC_EXT.1.14 The TSF shall only establish a trusted channel if the presented identifier in the received certificate matches the configured reference identifier, where the presented and reference identifiers are of the following fields and types: [selection: *SAN: IP address, SAN: Fully Qualified Domain Name (FQDN), SAN: user FQDN, CN: IP address, CN: Fully Qualified Domain Name (FQDN), CN: user FQDN, Distinguished Name (DN)*] and [selection: *no other reference identifier type, [assignment: other supported reference identifier types]*].

5.1.2 FCS_RBG_EXT.1 Random Bit Generation

Family Behaviour
Components in this family address the requirements for random bit/number generation. This is a new family defined for the FCS class.

Component levelling

FCS_RBG_EXT.1 Random Bit Generation requires random bit generation to be performed in accordance with selected standards and seeded by an entropy source.

Management: FCS_RBG_EXT.1
The following actions could be considered for the management functions in FMT:
a) There are no management activities foreseen

Audit: FCS_RBG_EXT.1
The following actions should be auditable if FAU_GEN Security audit data generation is included in the PP/ST:
a) Minimal: failure of the randomization process

FCS_RBG_EXT.1 Random Bit Generation
Hierarchical to: No other components

Dependencies: No other components

FCS_RBG_EXT.1.1 The TSF shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using [selection: *Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)*].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from [selection: *[assignment: number of software-based sources] software-based noise source, [assignment: number of platform-based sources] platform-based noise source*] with a minimum of [selection: *128 bits, 192 bits, 256 bits*] of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 “Security Strength Table for Hash Functions”, of the keys and hashes that it will generate.

5.2 Extended security components rationale

The new FCS_IPSEC_EXT family and its components FCS_IPSEC_EXT.1 – FCS_IPSEC_EXT.14 used in this ST are introduced by the cPPND [6].

The new FCS_RBG_EXT family and the FCS_RBG_EXT.1 component as used in this ST is introduced by the cPPND to satisfy a dependency on the part of the extended component FCS_IPSEC_EXT.1.

6 Security requirements for the TOE

This section descriptions the security functional and assurance requirements for the TOE. The notation, formatting, and conventions used are defined below.

Application notes have been added by the ST authors to provide the reader with additional understanding or to clarify the ST author's intent; they are italicized and usually appear immediately preceding or following the element needing clarification.

CC component operations are identified in the following way:

- Assignments and Selections are typeset in **bold text**
- Refinements are identified by **bold underlined** text for any additions and by ~~strikes thru~~ for any removals; any refinement that performs a deletion in text is also noted in a numbered footnote.

6.1 Security functional requirements for the TOE

This section describes the security functional requirements for the TOE. The security functional requirement components in this security target are CC Part 2 conformant or CC Part 2 extended.

Typographical and notational conventions introduced above are in reference to operations applied in the ST on the SFRs.

A summary of the security functional requirements is given by Table 8.

Table 8 – TOE Security Functional Requirements Summary

SFR	Description
FAU_GEN.1	Security audit data generation
FCS_CKM.2	Cryptographic key distribution (refined to key establishment)
FCS_CKM.4	Cryptographic key destruction
FCS_IPSEC_EXT.1	IPsec protocol
FPT_TEE.1	Testing of external entities
FPT_TST.1	TSF (self) testing
FTP_ITC.1	Inter-TSF trusted channel

6.1.1 Security Audit (FAU)

6.1.1.1 Security audit data generation (FAU_GEN)

6.1.1.1.1 Audit data generation (FAU_GEN.1)

FAU_GEN.1.1 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the **not specified** level of audit; and

c) **The auditable events: TOE start-up, TOE shutdown, TOE self-test failure, strongSwan external entity test failure, IKE SA initiation, IKE SA establishment, IKE SA reinitiation (rekeying), IKE SA reauthentication, IKE SA deletion, IKE CHILD SA establishment, IKE CHILD SA expiration.**

Application note: *From the standpoint of the TOE, the start-up and shutdown of the audit functions shall correspond to the start-up and shutdown of the TOE.*

FAU_GEN.1.2 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity (if applicable), and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the ST, **no other information**.

6.1.2 Cryptographic Support (FCS)

6.1.2.1 Cryptographic key management (FCS_CKM)

6.1.2.1.1 Cryptographic key ~~distribution~~ **establishment** (FCS_CKM.2)

FCS_CKM.2.1 The TSF shall ~~distribute~~ **perform** cryptographic ~~keys~~ **key establishment** in accordance with a specified cryptographic key ~~distribution~~ **establishment** method **IKEv2 as defined in RFC 7296, with**:

- **Key establishment schemes using Diffie-Hellman MODP groups that meet the following: RFC 3526**
- **Key establishment schemes using Diffie-Hellman ECP groups that meet the following: RFC 5114**
- **Key establishment schemes using Diffie-Hellman ECP groups that meet the following: RFC 6954**

~~that meets the following: [assignment: list of standards]~~.

Application note: *The distribution of cryptographic keys is done as part of the IPSEC and IKEv2 protocol that defines the key agreement between the client and the server. The TOE acts as a IPSEC client in a scenario where it obtains data from a non-TOE component. The cryptographic operation is done as part of FCS_COP.1 implemented by the platform. The kernel implementation is part of the Linux Kernel 4.19 and accessible using the Linux Kernel Crypto API. The user space crypto is implemented by OpenSSL 1.1.1b and accessible using the SoftHSM 2.6.1.*

6.1.2.1.2 Cryptographic key destruction (FCS_CKM.4)

FCS_CKM.4.1	The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method **overwriting with zeroes** that meets the following: **none**.
Application note:	*The MVC TOE destroys session keys stored in user space memory by overwriting with zeroes. Long-term keys are managed by the platform. Cryptographic keys in kernel space are erased by memory management of the kernel and part of the platform as described in OE.PT_CRYPO_EXP.*

6.1.2.2 Internet Protocol Security (IPsec) Extended (FCS_IPSEC_EXT)

The IPsec protocol is used to establish a mutually authenticated communication channel between subjects that provides data confidentiality and integrity. IPsec is a peer-to-peer protocol and as such does not need to be separated into distinct client and server requirements.

Application note:	*IPsec Protocol Security Functional Requirement extended component definitions are taken from the collaborative Protection Profile for Network Devices* [6]*.*

6.1.2.2.1 IPsec Protocol (FCS_IPSEC_EXT.1)

FCS_IPSEC_EXT.1.1	The TSF shall implement the IPsec architecture as specified in RFC 4301.
Application note:	*RFC 4301 calls for an IPsec implementation to use a Security Policy Database (SPD) to define how IP packets are to be handled.*
FCS_IPSEC_EXT.1.2	The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched and discards it.
FCS_IPSEC_EXT.1.3	The TSF shall implement **tunnel mode**.
FCS_IPSEC_EXT.1.4	The TSF shall implement the IPsec protocol ESP as defined in RFC 4303 using cryptographic algorithms **AES-CBC-128 (RFC3602), AES-CBC-256 (RFC3602), AES-GCM-128 (RFC 4106), AES-GCM-192 (RFC 4106), AES-GCM-256 (RFC 4106)** together with a Secure Hash Algorithm (SHA)-based HMAC **HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512**.
FCS_IPSEC_EXT.1.5	The TSF shall implement the protocol: **IKEv2 as defined in RFC 7296[1] and with mandatory support for NAT traversal as specified in RFC 7296[2], section 2.23, and RFC 4868 for hash functions.**
Application note:	*IPsec Protocol Security Functional Requirements are adapted from the collaborative Protection Profile for Network Devices revision v2.2e. The obsoleted RFC 5996 in FCS_IPSEC_EXT.1.5 has been upgraded to RFC 7296.*
FCS_IPSEC_EXT.1.6	The TSF shall ensure the encrypted payload in the **IKEv2** protocol uses the cryptographic algorithms **AES-CBC-128, AES-CBC-256 (specified in RFC 3602), AES-GCM-128, AES-GCM-192, AES-GCM-256 (specified in RFC 5282)**.
FCS_IPSEC_EXT.1.7	The TSF shall ensure that **IKEv2 SA lifetimes can be configured by a Security Administrator based on:** • **number of bytes*;***

[1] Reference to RFC 5669 has been replaced with a reference to RFC 7296 [r2]
[2] Reference to RFC 5669 has been replaced with a reference to RFC 7296 [r2]

- **length of time, where the time values can be configured within 1-48 hours.**

FCS_IPSEC_EXT.1.8 The TSF shall ensure that **IKEv2 Child SA lifetimes can be configured by a Security Administrator based on:**

- **number of bytes;**
- **length of time, where the time values can be configured within 1-24 hours.**

FCS_IPSEC_EXT.1.9 The TSF shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least **224, 256, 384, 512** bits.

Application note: *The MVC TOE does not contain the implementation of the cryptographic algorithms, so FCS_RBG_EXT.1 is addressed by OE.PT_CRYPTO_EXP.*

FCS_IPSEC_ETX.1.10 The TSF shall generate nonces used in **IKEv2** exchanges of length **according to the security strength associated with the negotiated Diffie-Hellman group.**

FCS_IPSEC_EXT.1.11 The TSF shall ensure that IKE protocols implement DH Group(s):

- **14 (2048-bit MODP), 15 (3072-bit MODP), 16 (4096-bit MODP) according to RFC 3526,**
- **19 (256-bit Random ECP), 20 (384-bit Random ECP), 21 (521-bit Random ECP) according to RFC 5114.**
- **25 (192-bit Random ECP), 26 (224-bit Random ECP) according to RFC 5114.**
- **27 (Brainpool 224-bit Random ECP), 28 (Brainpool 256-bit Random ECP), 29 (Brainpool 384-bit Random ECP), 30 (Brainpool 512-bit Random ECP) according to RFC 6954.**

FCS_IPSEC_EXT.1.12 The TSF shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the **IKEv2 IKE_SA** connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the **IKEv2 CHILD_SA** connection.

FCS_IPSEC_EXT.1.13 The TSF shall ensure that all IKE protocols perform peer authentication using **RSA, ECDSA** that use X.509v3 certificates that conform to RFC 4945 and **no other method**.

FCS_IPSEC_EXT.1.14 The TSF shall only establish a trusted channel if the presented identifier in the received certificate matches the configured reference identifier, where the presented and reference identifiers are of the following fields and types: **SAN: IP address, SAN: Fully Qualified Domain Name (FQDN), SAN: user FQDN, Distinguished Name (DN)** and **no other reference identifier type**.

6.1.3 Protection of the TSF (FPT)

6.1.3.1 Testing of external entities (FPT_TEE)

6.1.3.1.1 Testing of external entities (FPT_TEE.1)

FPT_TEE.1.1 The TSF shall run a suite of tests **during initial start-up** and **under no other conditions** to check the fulfillment of **correct operation of the cryptographic algorithms, and correct operation of the random bit generator**.

FPT_TEE.1.2	If the test fails, the TSF shall **refuse to start or disable the failing algorithm**.
Application note:	*Since the MVC TOE does not implement the cryptographic algorithms, the start-up test of the cryptographic algorithms is addressed by the testing of external entities. The kernel is configured to panic and halt the system if a kernel space cryptographic algorithm test fails. If a user space cryptographic algorithm test fails, the TOE refuses to load the failing algorithm thus preventing use of a misbehaving implementation.*

6.1.3.2 TSF self test (FPT_TST)

6.1.3.2.1 TSF testing (FPT_TST.1)

FPT_TST.1.1	The TSF shall run a suite of self tests **during initial start-up** and **under no other conditions** to demonstrate the correct operation of **parts of the TSF by verifying the strongSwan configuration data and parts of the TSF executables**.
FPT_TST.1.2	The TSF shall provide authorised users with the capability to verify the integrity of **no parts of the TSF data**.
FPT_TST.1.3	The TSF shall provide authorised users with the capability to verify the integrity of **no parts of the TSF**.
Application note:	*The TOE performs the verification of configuration data and parts of TSF executables during startup.*

6.1.4 Trusted path/channels (FTP)

6.1.4.1 Inter-TSF trusted channel (FTP_ITC)

6.1.4.1.1 Inter-TSF trusted channel (FTP_ITC.1)

FTP_ITC.1.1	The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.
FTP_ITC.1.2	The TSF shall permit **the TSF** to initiate communication via the trusted channel.
FTP_ITC.1.3	The TSF shall initiate communication via the trusted channel for **exchange of red data**.
Application note:	*Red data is not to be sent or received without encryption to protect confidentiality and integrity. Any red data sent through the TOE to a VPN peer or received from a peer goes through the trusted channel as is established by FCS_IPSEC_EXT.1 as an IPsec tunnel.*

6.1.5 Security functional requirements provided by the platform

The following cryptographic SFRs have been added because of the CSEC Scheme Crypto Policy 188 [12].

6.1.5.1 Cryptographic key management (FCS_CKM)

6.1.5.1.1 Cryptographic Key Generation (FCS_CKM.1)

FCS_CKM.1.1	The ~~TSF~~**PLT** shall generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm:

- **ECC schemes using 'NIST curves' P-192, P-224, P-256, P-384, P-521 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4;**
- **FFC schemes using cryptographic key sizes of 2048, 3072 and 4096-bit that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1;**
- **ECC schemes using 'Brainpool curves' brainpoolP224r1, brainpoolP256r1, brainpoolP384r1 and brainpoolP512r1 that meet the following: RFC 5639**

~~and specified cryptographic key sizes [assignment: *cryptographic key sizes*] that meet the following: [assignment: *list of standards*].~~

Application note: *This SFR specifies the generation of Diffie-Hellman values that are required fo the Diffie-Hellman key exchanges specified in FCS_CKM.2*

6.1.5.2 Cryptographic operation (FCS_COP)

6.1.5.2.1 Cryptographic operation (FCS_COP.1)

FCS_COP.1.1/DataEncryption The ~~TSF~~**PLT** shall perform **encryption/decryption** in accordance with a specified cryptographic algorithm **AES used in CBC and GCM mode** and cryptographic key sizes **128 bits, 192 bits and 256 bits** that meet the following: **AES as specified in ISO 18033-3, CBC as specified in ISO 10116 and GCM as specified in ISO 19772**.

Application note: *The cryptographic primitives are all performed by the OpenSSL library, which is part of the operational environment.*

FCS_COP.1.1/SigGen The ~~TSF~~**PLT** shall perform **cryptographic signature services (generation and verification)** in accordance with a specified cryptographic algorithm

- **RSA Digital Signature Algorithm and cryptographic key sizes (modulus) 2048, 3072 and 4096,**
- **Elliptic Curve Digital Signature Algorithm and cryptographic key sizes 256 bits, 384 bits, 512 bits and 521 bits.**

~~and cryptographic key sizes [assignment: cryptographic key sizes]~~ that meet the following:

- **For RSA schemes: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3,**
- **For ECDSA schemes: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" P-256, P-384 and P-521; ISO/IEC 14888-3, Section 6.4**
- **For ECDSA schemes using P-256, P-384 and P-521 brainpool curves: RFC 5639**

FCS_COP.1.1/Hash The ~~TSF~~**PLT** shall perform **cryptographic hashing services** in accordance with a specified cryptographic algorithm **SHA-256, SHA-384 and SHA-512** ~~and cryptographic key sizes [assignment: cryptographic key sizes]~~ **and**

	message digest sizes 256, 384 and 512 bits that meet the following: **ISO/IEC10118-3:2004**.
FCS_COP.1.1/KeyedHash	The ~~TSF~~**PLT** shall perform **keyed-hash message authentication** in accordance with a specified cryptographic algorithm **HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512** and cryptographic key sizes **256, 384 and 512 bits and message digest sizes 256, 384 and 512 bits** that meet the following**: ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"**.
Application note:	*The cryptographic operation is done as part of FCS_COP.1 implemented by the platform. The kernel implementation is part of the Linux Kernel 4.19 and accessible using the Linux Kernel Crypto API. The user space crypto is implemented by OpenSSL 1.1.1b and accessible using the SoftHSM 2.6.1.*

6.1.5.3 Random Bit Generation (FCS_RBG_EXT)

6.1.5.3.1 Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1	The ~~TSF~~**PLT** shall perform all deterministic random bit generation services in accordance with ISO/IEC 18031:2011 using **HMAC_DRBG (any)**.
FCS_RBG_EXT.1.2	The deterministic RBG shall be seeded by at least one entropy source that accumulates entropy from **one platform-based noise source** with a minimum of **256 bits** of entropy at least equal to the greatest security strength, according to ISO/IEC 18031:2011 Table C.1 "Security Strength Table for Hash Functions", of the keys and hashes that it will generate.
Application note:	*The TOE relies on the pseudo random generator provided by OpenSSL 1.1.1b, which is part of the operational environment.*

6.1.6 Security functional requirements rationale

The TOE is a VPN client used to establish a cryptographically secure data communication channel between a local user and a remote trusted user or to establish a trustworthy virtual network over a potentially unsafe network. The trustworthy virtual network must be suitable to securely transfer red data. The VPN client is physically realized in the MVC TOE as a software product (represented by Variant 2 in [11]). The core functionality is to transmit red data with confidentiality, integrity and authenticity by establishing a suitably configured VPN tunnel between the VPN client and a VPN gateway or other network device implementing the same protocol. This TOE is limited, in terms for the functionality provided, to that service.

Several sources were consulted to identify the necessary TOE functional requirements, including [6] and [11]. Supporting functions are provided by the platform (PLT).

6.1.6.1 Security functional requirements rationale

Table 9 demonstrates that all of the security objectives for the TOE are met by the SFRs and the following rationale describes how the objectives map to the SFRs:

Table 9 – Mapping of Objectives to SFRs										
	FAU_GEN.1	FCS_CKM.1	FCS_CKM.2	FCS_CKM.4	FCS_COP.1	FCS_IPSEC_EXT.1	FCS_RBG_EXT.1	FPT_TST.1	FPT_TEE.1	FTP_ITC.1

O.COMMUNICATION			X	X		X				
O.AUDIT	X									
O.SELF_TEST								X	X	
O.INFO_FLOW						X				X
OE.PT_CRYPTO_EXP		X			X		X			

Note: Although the crypto graphic operations FCS_CKM.1, FCS_COP.1 and FCS_RGB_EXT.1 are all provided by the operational environment the CSEC crypto policy SP188 requires them to be specified as SFRs in the ST. So for completeness they are also listed here in this table.

6.1.6.1.1 Rationale for SFRs meeting O.COMMUNICATION

O.COMMUNICATION is the primary objective of the TOE (intended to counter the primary threat T.NETWORK_ATTACK). The threat encompasses all of the manipulations that can be performed on the black network by a powerful attacker operating within the symbolic model. The objective is realized by the implementation of the IPsec architecture, protocols, and cryptographic algorithms represented by the various elements of FCS_IPSEC_EXT.1. The protocols achieve the establishment of a trusted channel that is mutually authenticated, provides confidentiality and integrity of red data exchanged over the channel, and is resistant to network message manipulation attacks. Mechanisms meeting this objective must additionally provide the distribution of negotiated keys to the communication participants through the operation of the IPsec and IKEv2 protocols implemented by the TOE as required by FCS_CKM.2, and confidentiality of user data including data used as cryptographic keys in cryptographic operations employed by the protocols, which shall not retain them beyond their required use and shall be destroyed as required by FCS_CKM.4.

6.1.6.1.2 Rationale for SFRs meeting O.AUDIT

The objective to generate security-relevant events is achieved by FAU_GEN.1, stating that audit events will be generated for start-up, shutdown, self- test failure, external entity test failure and a number of IKE events.

6.1.6.1.3 Rationale for SFRs meeting O.SELF_TEST

The objective to perform self tests to detect malfunction of security-enforcing or security-supporting functions is achieved by FPT_TST.1 and FPT_TEE.1. The TOE will verify the integrity of the configuration and verify that the cryptographic operations are performed correctly before any outside connections is established. If the tests are failing the TOE will generate an audit record and refuse to start or disable the failing algorithm.

6.1.6.1.4 Rationale for SFRs meeting O.INFO_FLOW

The objective to prevent modification or disclosure of red data is achieved by FCS_IPSEC_EXT.1, which provides the protocols and mechanisms for mutual authentication and secure communication, and FTP_ITC.1, which provides for the establishment of the trusted channel for exchange of red data with the peer.

6.1.6.1.5 Rationale for SFRs meeting OE.PT_CRYPTO_EXP

The objective for the environment platform to provide the explicitly specified cryptographic primitives is achieved by the SFRs satisfied by the platform as enumerated in Section 6.1.5: FCS_CKM.1, FCS_COP.1, and FCS_RBG_EXT.1. These SFRs also satisfy the dependencies arising from FCS_IPSEC_EXT.1.

6.1.6.2 SFR dependencies

Table 10 shows the dependencies for each of the SFRs and how they are resolved. Note that in the dependencies column that alternative dependencies (such as in FCS_CKM.4) are represented in a single cell whereas multiple dependencies (such as in FCS_IPSEC_EXT.1) are represented in distinct cells.

Table 10 – SFR Dependencies		
TOE SFR	Dependencies	How dependency is resolved
FAU_GEN.1 Audit data generation	FPT_STM.1	Not resolved by FPT_STM.1. Timestamps are provided as part of the platform objective OE.PT_TIME
FCS_CKM.2 Cryptographic key distribution	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	Satisfied by FCS_CKM.1
	FCS_CKM.4	Satisfied by FCS_CKM.4
FCS_CKM.4 Cryptographic key destruction	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	Satisfied by FCS_CKM.1 for keys generated by the platform and by FCS_CKM.2 for session keys negotiated with the IPSEC channel.
FCS_IPSEC_EXT.1 IPsec protocol	FCS_CKM.1 Cryptographic key generation	Satisfied FCS_CMK.1
	FCS_CKM.2 Cryptographic key establishment	Satisfied FCS_CMK.2
	FCS_COP.1/DataEncryption Cryptographic Operation (AES Data encryption/decryption)	Satisfied FCS_COP.1/ DataEncryption Cryptographic Operation (AES Data encryption/decryption),
	FCS_COP.1/SigGen Signature Generation Cryptographic operation	Satisfied FCS_COP.1/SigGen Signature Generation Cryptographic operation
	FCS_COP.1/Hash Cryptographic operation (Hash Algorithm)	Satisfied by FCS_COP.1/Hash Cryptographic operation (Hash Algorithm)
	FCS_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)	Satisfied by FCS_COP.1/KeyedHash Cryptographic operation (Keyed Hash Algorithm)
	FCS_RBG_EXT.1 Random Bit Generation	Satisfied by FCS_RBG_EXT.1 Random Bit Generation.
FPT_TEE.1 Testing of external entities	No dependencies	-
FPT_TST.1 TSF (self) testing	No dependencies	-
FTP_ITC.1 Inter-TSF trusted channel	No dependencies	-
Non-TOE SFR	Dependencies	How dependency is resolved
FCS_CKM.1	FCS_CKM.2 or FCS_COP.1	Satisfied by FCS_CKM.2
	FCS_CKM.4	Satisfied by FCS_CKM.4
FCS_COP.1.1/DataEncryption	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	Satisfied by FCS_CKM.2
	FCS_CKM.4	Satisfied by FCS_CKM.4
FCS_COP.1.1/SigGen	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	No, but satisfied by OE.PT_CRYPTO_GEN that ensure that RSA and EC keys are available for the use of the TOE.
	FCS_CKM.4	No, the long term keys are protected by OE.PT_CRYPTO_GEN
FCS_COP.1.1/Hash	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	The dependency is not resolved since no key is required in a Hash operation.
	FCS_CKM.4	The dependency is not resolved since no key is required in a Hash operation.
FCS_COP.1.1/KeyedHash	FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1	Satisfied by FCS_CKM.2

Table 10 – SFR Dependencies		
TOE SFR	Dependencies	How dependency is resolved
	FCS_CKM.4	Satisfied by FCS_CKM.4
FCS_RBG_EXT.1	No dependencies	-

6.2 Security assurance requirements for the TOE

The security assurance requirements of the Security Target are those defined for assurance level EAL4 augmented with ALC_FLR.2 and AVA_VAN.4. The relevant SARs are summarized in Table 11.

Table 11 – TOE Security Assurance Requirements Summary EAL4+		
Assurance Class	Assurance Component	Assurance Component Description
Development	ADV_ARC.1	Security architecture description
	ADV_FSP.4	Complete functional specification
	ADV_IMP.1	Implementation representation of the TSF
	ADV_TDS.3	Basic modular design
Guidance Documents	AGD_OPE.1	Operational user guidance
	AGD_PRE.1	Preparative procedures
Life-cycle Support	ALC_CMC.4	Production support, acceptance procedures and automation
	ALC_CMS.4	Problem tracking CM coverage
	ALC_DEL.1	Delivery procedures
	ALC_DVS.1	Identification of security measures
	ALC_FLR.2	Flaw reporting procedures
	ALC_LCD.1	Developer defined life-cycle model
	ALC_TAT.1	Well-defined development tools
Tests	ATE_COV.2	Analysis of Coverage
	ATE_DPT.1	Testing: basic design
	ATE_FUN.1	Functional testing
	ATE_IND.2	Independent testing - sample
Vulnerability Assessment	AVA_VAN.4	Methodical vulnerability analysis

6.2.1 Security assurance requirements rationale

The integrator for the TOE intends to incorporate the TOE in systems that will process at most VS NfD “restricted” information and has specified the EAL4+ assurance level. The Integrator’s choice of EAL4 conformance and its augmentation is based on a precedent established by BSI of Germany. Specifically, these assurance requirements for a VPN client are based on the VS-Anforderungsprofil VPN-Client (VPNC), 19.09.2018 [11].

7 TOE Summary Specification

7.1 Mapping of security features from the TOE description to SFRs

Table 12 presents a mapping from security features presented in Section1.4.5 to the SFRs identified in Section 6.1.

Table 12 – Mapping of Security Features from the TOE Description to SFRs

	FAU_GEN.1	FCS_CKM.1 (PLT)	FCS_CKM.2	FCS_CKM.4	FCS_COP.1 (PLT)	FCS_IPSEC_EXT.1	FCS_RBG_EXT.1 (PLT)	FPT_TEE.1	FPT_TST.1	FTP_ITC.1
1. Cryptographic protocols (and operations)				X	X	X	X			
2. Cryptographic key negotiation and handling		X	X							
3. Establish mutually-authenticated trusted channel (VPN tunnel)										X
4. Confidentiality and integrity of red data-in-transit						X				
5. Generation of audit records	X									
6. Self-test of the cryptographic functions								X	X	

7.2 Security audit

7.2.1 Security audit data generation (FAU_GEN)

Security audit data is generated by strongSwan. Audit data (logs) are written to the system journal and can be accessed with systemd tools. The TSF can also be configured to write logs to a file, or to pass logs to the syslog(3) POSIX function.

StrongSwan allows the log level to be configured. Logging can be turned completely off and can be configured to log sensitive material such as keys.

Storing logs and maintaining their integrity and confidentiality is a responsibility of the PLT.

Table 13 – FAU implementation

SFR	Summary of SFR implementation
FAU_GEN.1.1	The TOE generates audit records for the following events: • IKE SA initiation • IKE SA establishment • IKE SA reinitiation (rekeying) • IKE SA reauthentication • IKE SA deletion • IKE CHILD SA establishment • IKE CHILD SA expiration • Failures in external cryptographic algorithm tests • Failure in entropy test

Table 13 – FAU implementation	
SFR	Summary of SFR implementation
FAU_GEN.1.2	Generated audit events contain following information: • Date and time of the event • Type of event • Subject identity (if applicable) • The outcome (success or failure) of the event

7.3 Cryptographic support

7.3.1 Cryptographic key management (FCS_CKM)

The TOE implements FCS_CKM.2 Cryptographic Key Establishment and FCS_CKM.4 Cryptographic Key Destruction (for session keys). The functions FCS_CKM.1 Cryptographic Key Generation is implemented by the platform (see the tables below).

Below is a table with cryptographic algorithms used within the TOE, its purpose, algorithm and the component that provides the cryptographic operation.

Table 14 – Cryptographic algorithms		
Purpose	Algorithm(s)	Provider
ESP / Encryption	*AES-CBC-128* *AES-CBC-256*	Kernel Crypto API
ESP / Authentication	*HMAC-SHA-256* *HMAC-SHA-384* *HMAC-SHA-512*	Kernel Crypto API
ESP / Authenticated Encryption	*AES-GCM-128* *AES-GCM-192* *AES-GCM-256*	Kernel Crypto API
IKEv2 / Encryption	*AES-CBC-128* *AES-CBC-256*	OpenSSL
IKEv2 / Authentication	*HMAC-SHA-256* *HMAC-SHA-384* *HMAC-SHA-512*	OpenSSL
IKEv2 / Authenticated Encryption	*AES-GCM-128* *AES-GCM-192* *AES-GCM-256*	OpenSSL
IKEv2 / DH Key Exchange	*2048-bit MODP* *3072-bit MODP* *4096-bit MODP* *256-bit Random ECP* *384-bit Random ECP* *521-bit Random ECP* *NIST 192-bit Random ECP* *NIST 224-bit Random ECP* *Brainpool 224-bit Random ECP* *Brainpool 256-bit Random ECP* *Brainpool 384-bit Random ECP* *Brainpool 512-bit Random ECP*	OpenSSL
IKEv2 / Peer Authentication	*RSA (2048, 3072, 4096 bits)* *ECDSA (192, 224, 256, 384, 512, 521 bits)*	OpenSSL through SoftHSM
Disk encryption (*)	*AES-CBC-ESSIV (256 bits)* *AES-XTS (256 bits)*	Kernel Crypto API

Table 14 – Cryptographic algorithms		
Purpose	Algorithm(s)	Provider
Disk encryption key wrapping (*)	*RSA (2048)* *ECC (BN256, P256)*	TPM

The purposes marked by (*) above are not used by the TOE for any security functions of to meet any security objectives. They are only mentioned here for completeness. The provider versions used are OpenSSL 1.1.1b, SoftHSM 2.6.1 and Linux Kernel 4.19.

The following table list all the cryptographic keys that are used within the TOE. The table shows for each key the types, context, origin storage and destruction.

Table 15 – Cryptographic keys					
Key	Type	Context	Origin	Storage	Destruction
ESP Encryption key	*AES_CBC*	*TOE*	*Established by IKEv2 (CREATE_CHILD_SA)*	*Plaintext in RAM*	Zeroized when no longer used
ESP Authentication key	*HMAC_SHA2*	*TOE*	*Established by IKEv2 (CREATE_CHILD_SA)*	*Plaintext in RAM*	Zeroized when no longer used
ESP Encryption+Authentication key	*AES_GCM*	*TOE*	*Established by IKEv2 (CREATE_CHILD_SA)*	*Plaintext in RAM*	Zeroized when no longer used
IKEv2 encryption key	*AES_CBC*	*TOE*	*Established by IKEv2 (IKE_SA_INIT)*	*Plaintext in RAM*	Zeroized when no longer used
IKEv2 authentication key	*HMAC_SHA2*	*TOE*	*Established by IKEv2 (IKE_SA_INIT)*	*Plaintext in RAM*	Zeroized when no longer used
IKEv2 encryption+authentication key	*AES_GCM*	*TOE*	*Established by IKEv2 (IKE_SA_INIT)*	*Plaintext in RAM*	Zeroized when no longer used
IKEv2 DH Key Exchange parameters	*MODP, ECP*	*PLT*	*Generated by OpenSSL*	*Plaintext in RAM*	Zeroized when no longer used
IKEv2 / Peer Authentication keypair	*RSA, ECDSA*	*PLT*	*Generated on SoftHSM or imported into SoftHSM*	*Plaintext in RAM, encrypted in flash*	Zeroized in RAM when no longer used, zeroized in flash when removed
IKEv2 / Peer Authentication certificate	*RSA, ECDSA*	*PLT*	*Imported into SoftHSM*	*Plaintext in RAM, encrypted in flash*	Zeroized in RAM when no longer used, zeroized in flash when removed
TPM primary key	*RSA, ECC*	*PLT*	*Generated on TPM*	*In TPM*	Managed by TPM
SoftHSM data encryption key	*passphrase*	*PLT*	*Generated from /dev/random*	*Plaintext in RAM, sealed in TPM*	Managed by TPM

7.3.2 Internet protocol security (IPsec) extended (FCS_IPSEC_EXT)

This function is used to implement the trusted path with the VPN peer.

MVC implements the IPsec protocol architecture as specified in RFC 4301. Implementation is based on strongSwan and Linux kernel XFRM functionality.

StrongSwan implements IKEv2 as specified in RFC 7296 (updating RFC 5669) and manages the Security Policy Database (SPD), and IKE and IPsec security associations (SAs). StrongSwan communicates the SPD and SAs to the XFRM kernel module which in turn is responsible for enforcing the contents of the SPD and performing encryption/decryption on packets sent to, or received from, an IPsec tunnel according to the defined SAs.

StrongSwan communicates the SPD and SAs to the XFRM module via the Netlink socket using NETLINK_XFRM protocol. Netlink socket implementation is part of the PLT.

StrongSwan configuration is performed via the VICI-plugin. Modifying the configuration is performed via a Unix domain socket connection to the IKE daemon. Accessing the configuration socket is allowed only for security administrators using the PLT access control policy enforcement.

StrongSwan and XFRM implementations are broader in scope than MVC. Allowed configuration options for MVC are restricted.

Cryptographic algorithm implementations are part of the PLT. Correct operation of cryptographic primitives supplied by the PLT and used by the TSF is verified as described in Section 7.4

Table 16 – FCS implementation

SFR	Summary of SFR implementation
FCS_CKM.2	The TOE performs key establishment as part of the IPSEC and IKEv2 protocol.
FCS_CKM.4	The TOE destroys all session key material in user space, derived by execution of the IPsec/IKE protocols, in its memory after use by overwriting with zeroes. Any session keys in kernel space are erased by the kernel memory management of the platform as described in OE.PT_CRYPO_EXP.
FCS_IPSEC_EXT.1.1	The TOE implements IPsec architecture as specified in RFC 4301. The TOE supports both IPv4 and IPv6.
FCS_IPSEC_EXT.1.2	The TOE drops all packets not matching an SPD entry.
FCS_IPSEC_EXT.1.3	The TSF implements IPsec tunnel mode.
FCS_IPSEC_EXT.1.4	The TSF implements ESP. Cryptographic algorithms are provided by the platform. The TSF truncates the output MAC value of HMAC-SHA-256 algorithm to 128 bits, HMAC-SHA-384 to 192 bits, and HMAC-SHA 512 to 256 bits. For compatibility reasons, HMAC-SHA-256 truncation can be overridden in configuration by setting it to 96 bits.
FCS_IPSEC_EXT.1.5	The TOE implements IKEv2 for endpoint authentication and key exchange.
FCS_IPSEC_EXT.1.6	The TOE uses AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-192, AES-GCM-256 algorithms provided by the platform.

Table 16 – FCS implementation	
SFR	Summary of SFR implementation
	The TOE uses HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 algorithms provided by the platform.
FCS_IPSEC_EXT.1.7	The TOE implements support for configuring IKEv2 SA lifetimes via a programmatic interface. Lifetimes can be configured in length of time or number of bytes. Use of configuration interface is limited to platform-defined security administrators only.
FCS_IPSEC_EXT.1.8	The TOE implements support for configuring IKEv2 Child SA lifetimes via a programmatic interface. Lifetimes can be configured in length of time or number of bytes. Use of configuration interface is limited to platform-defined security administrators only.
FCS_IPSEC_EXT.1.9	The TOE uses platform provided RBG to generate DH parameters.
FCS_IPSEC_EXT.1.10	The TOE generates 32-byte nonces for IKEv2 exchanges using platform-provided RBG.
FCS_IPSEC_EXT.1.11	The TOE implements support for DH groups: regular groups 14, 15, 16; NIST Elliptic Curve Groups 19, 20, 21, 25, 26; Brainpool Elliptic Curve Groups 27, 28, 29, 30.
FCS_IPSEC_EXT.1.12	The TOE configuration guidance in the CGX 2.6 Getting Started Guide provides instructions on configuring IKEv2 and ESP proposals in such way that chosen IKEv2 IKE_SA is of equal or greater strength than chosen IKEv2 CHILD_SA.
FCS_IPSEC_EXT.1.13	The TOE implements X.509v3 support for peer authentication.
FCS_IPSEC_EXT.1.14	The TOE authenticates peers using X.509v3 certificates. Following certificate fields are supported for inferring peer identity: • SAN: IP address • SAN: Fully Qualified Domain Name (FQDN) • SAN: user FQDN • Distinguished Name (DN)

Table 17 – FCS_IPSEC_EXT.1 dependencies satisfied by the platform	
SFR	Summary of SFR dependency
FCS_CKM.1	StrongSwan uses user space OpenSSL library provided by the platform to generate Diffie-Hellmann values for key establishments within IKEv2.
FCS_COP.1/DataEncryption	To encrypt and decrypt ESP messages the XFRM kernel module calls kernel crypto API provided by the platform. User space StrongSwan uses cryptographic algorithms from OpenSSL library for IKEv2 protocol encryption.
FCS_COP.1/SigGen	StrongSwan uses PKCS#11 module/OpenSSL library to perform X.509v3 certificate signature generation/verification in IKEv2 negotiation.
FCS_COP.1/Hash	StrongSwan uses OpenSSL to perform cryptographic hash calculation of IKEv2 signatures.

Table 17 – FCS_IPSEC_EXT.1 dependencies satisfied by the platform	
SFR	Summary of SFR dependency
FCS_COP.1/KeyedHash	The TOE uses HMAC functions of OpenSSL library to ensure integrity of IKEv2 messages. The TOE uses HMAC functions of Kernel Crypto API to protect ESP message integrity.
FCS_RBG_EXT.1	StrongSwan relies on OpenSSL on randomness generation. OpenSSL uses pseudo random generator provided by the core platform.

7.4 Protection of the TSF

7.4.1 Testing of external entities (FPT_TEE)

StrongSwan performs crypto tests on startup. The Linux kernel tests cryptographic primitives used by the XFRM module during the system boot. These are described further in Table 18.

7.4.2 TSF self test (FPT_TST)

The TSF self test comprises two parts: test of TSF components through FPT_TST and test of the external entities through FPT_TEE. The self test part is performed by verifying the integrity of the strongSwan configuration using SHA-256 a checksum, but also trigger the invocation of the tests performed through FPT_TEE.

The test of the external cryptographic functions that are invoked by the TSF, and therefore considered part of, the TSF start-up self test.

Table 18 – FPT implementation	
SFR	Summary of SFR implementation
FPT_TEE.1.1	The TSF tests proper operation for all cryptographic algorithms it uses that are provided by the platform. These tests are performed through a combination of system boot-time tests and strongSwan startup tests. The Linux kernel tests cryptographic primitives used by the XFRM module during system boot. The kernel space cryptographic tests include: • AES encrypt/decrypt for CBC mode from RFC 3602, NIST SP 800-38A • AES encrypt/decrypt for GCM mode from McGrew & Viega, The Galois/Counter Mode of Operation (GCM) • HMAC-SHA-256 from draft-ietf-ipsec-ciph-sha-256-01.txt • HMAC-SHA-384, HMAC-SHA-512 from RFC 4231 StrongSwan performs generic crypto tests on startup that can be run against any underlying crypto implementations. Each encryption is tested during demon initialization. Cryptographic primitives are

Table 18 – FPT implementation	
SFR	Summary of SFR implementation
	required to have at least one test vector defined. The tests cover crypto primitives used by strongSwan. Tests need to be enabled in the configuration. StrongSwan performs crypto tests for external cryptographic primitives during startup, including: • AES encrypt/decrypt for CBC mode from RFC 3602, NIST SP 800-38A • AES encrypt/decrypt for GCM mode from McGrew & Viega, The Galois/Counter Mode of Operation (GCM) • HMAC_DRBG from NIST SP 800-90A (DRBGVS) • Brainpool DH groups from RFC 6932 / RFC 7027 • DH groups from RFC 5114 • HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 from RFC 4868 • SHA-256, SHA-384, SHA-512 from The Secure Hash Algorithm Validation System, SHAVS
FPT_TEE.1.2	The kernel is configured (requiring a kernel command line parameter to be used) to panic and halt the system if a kernel space cryptographic algorithm test fails. If a user space cryptographic algorithm test fails, the TOE refuses to load the failing algorithm thus preventing use of a misbehaving implementation. The user space components of the MVC generate a log entry noting a failed test.
FPT_TST.1.1	At startup the MVC performs a set of self tests. At startup verification of the integrity of the parts of the strongSwan configuration and parts of the TSF included by the executable files listed below is done by calculating the checksums and comparing with previously stored checksums. For the self tests a simple integrity checker scheme verifies the SHA-256 checksums of a variety of files and will keep the strongSwan systemd service from starting if there is a mismatch in any of the checksums. As part of the build process, after the root file system has been created, a post process script is run as defined in the Bitbake class; integrity.bbclass. The first step is to insert the Simple Integrity script as defined by “CHKSUM_BIN” into the strongSwan systemd service. This will run as a systemd ExecStartPre function.

Table 18 – FPT implementation	
SFR	Summary of SFR implementation
	The checksums are created in three stages. • Checksum the systemd server itself • Checksum of all of the files defined in the variable "CHKSUM_FILE_LIST" is placed in the parent file "CHKSUM_FILE_NAME" • Checksum the file "CHKSUM_FILE_NAME" At startup time the systemd service will run the shell script defined by "CHKSUM_BIN" that performs three checks: • Verify the checksum of the systemd service itself (strongswan-swanctl.service) • Verify the "CHKSUM_FILE_NAME" file • Verify each file listed in "CHKSUM_FILE_LIST" If there is a mismatch in the checksum of any of these files the strongSwan service will not be started. The files included in the integrity check are: • /usr/lib/ipsec/libvici.so.0.0.0 • /usr/lib/ipsec/libcharon.so.0.0.0 • /usr/lib/ipsec/libstrongswan.so.0.0.0 • /usr/lib/ipsec/libvici.so.0.0.0 • /usr/bin/pki • /usr/sbin/charon-systemd • /usr/sbin/ipsec • /usr/sbin/swanctl • /usr/lib/ipsec/plugins/libstrongswan-aes.so • /usr/lib/ipsec/plugins/libstrongswan-resolve.so • /usr/lib/ipsec/plugins/libstrongswan-attr.so • /usr/lib/ipsec/plugins/libstrongswan-nonce.so • /usr/lib/ipsec/plugins/libstrongswan-revocation.so • /usr/lib/ipsec/plugins/libstrongswan-cmac.so • /usr/lib/ipsec/plugins/libstrongswan-openssl.so • /usr/lib/ipsec/plugins/libstrongswan-sha1.so • /usr/lib/ipsec/plugins/libstrongswan-constraints.so • /usr/lib/ipsec/plugins/libstrongswan-pem.so • /usr/lib/ipsec/plugins/libstrongswan-sha2.so • /usr/lib/ipsec/plugins/libstrongswan-pgp.so • /usr/lib/ipsec/plugins/libstrongswan-socket-default.so • /usr/lib/ipsec/plugins/libstrongswan-curl.so • /usr/lib/ipsec/plugins/libstrongswan-pkcs11.so • /usr/lib/ipsec/plugins/libstrongswan-sqlite.so • /usr/lib/ipsec/plugins/libstrongswan-des.so • /usr/lib/ipsec/plugins/libstrongswan-pkcs12.so

Table 18 – FPT implementation	
SFR	Summary of SFR implementation
	• /usr/lib/ipsec/plugins/libstrongswan-sshkey.so • /usr/lib/ipsec/plugins/libstrongswan-dnskey.so • /usr/lib/ipsec/plugins/libstrongswan-pkcs1.so • /usr/lib/ipsec/plugins/libstrongswan-stroke.so • /usr/lib/ipsec/plugins/libstrongswan-pkcs7.so • /usr/lib/ipsec/plugins/libstrongswan-updown.so • /usr/lib/ipsec/plugins/libstrongswan-gmp.so • /usr/lib/ipsec/plugins/libstrongswan-pkcs8.so • /usr/lib/ipsec/plugins/libstrongswan-vici.so • /usr/lib/ipsec/plugins/libstrongswan-hmac.so • /usr/lib/ipsec/plugins/libstrongswan-pubkey.so • /usr/lib/ipsec/plugins/libstrongswan-x509.so • /usr/lib/ipsec/plugins/libstrongswan-kernel-netlink.so • /usr/lib/ipsec/plugins/libstrongswan-random.so • /usr/lib/ipsec/plugins/libstrongswan-xauth-generic.so • /usr/lib/ipsec/plugins/libstrongswan-md5.so • /usr/lib/ipsec/plugins/libstrongswan-rc2.so • /usr/lib/ipsec/plugins/libstrongswan-xcbc.so
FPT_TST.1.2	See above.
FPT_TST.1.3	See above.

7.5 Trusted Channel

7.5.1 Inter-TSF trusted channel (FTP_ITC)

The MVC communicates red data with other compatible trusted peers over a trusted communication channel. This channel is established using the standard protocols for IPsec and IKEv2. All red data is forced flow only through the trusted channel. See Section 7.3.2 for a description of how the channel is established with IPsec.

The TOE uses keys and certificates stored in SoftHSM (the platform's encrypted certificate storage) via PKCS#11 API. Used certificates and keys are referred to in the IPsec configuration. All cryptographic operations are performed by OpenSSL and access from the TOE via PKCS#11 API of the SoftHSM,. The Configuration Agent (provided by the integrator) in Figure 1 is responsible for managing keys and certificates stored in SoftHSM.

Table 19 – FTP implementation	
SFR	Summary of SFR implementation
FTP_ITC.1.1	A mutually authenticated trusted channel is established with the VPN peer through use of the IPsec provided by FCS_IPSEC_EXT.1.4 and

Table 19 – FTP implementation	
SFR	Summary of SFR implementation
	IKEv2 protocols provided by FCS_IPSEC_EXT.1.5 as described in Table 16.
FTP_ITC.1.2	The VPN client initiates communication via the trusted channel with a compatible VPN peer.
FTP_ITC.1.3	All exchange of red data with the VPN peer occurs only via the trusted channel as initiated by the VPN client.

8 Glossary of terms and abbreviations

Table 20 – Terms	
Term	Definition
adversary	a person or other agent, not authorized to access the TOE or one of its peers but is able to exercise significant powers on the black network over which the TOE and its peers communicate, that mounts a network attack from the black network with the objective of violating one of the security policies
API	Application Programming Interface
ARM	Advanced RISC Machine (processor) (originally Acorn RISC Machine)
black network	a computer network that carries only cleartext nonsensitive data and encrypted red data (also referred to as an “untrusted network”) and that is accessible to adversaries
BSI	Bundesampt für Sicherheit in der Informationstechnik (Federal Office for Information Security)
CC	Common Criteria for Information Technology Security Evaluation Version 3.1r5
CBC	Cipher Block Chaining
CGL	Carrier Grade Linux (standard working group) CGL 5.0 standard
CGX	MontaVista Linux Carrier Grade eXpress operating system
charon-systemd	IKE daemon for use with systemd
cleartext	data that is not encrypted
COTS	Commercial-Off-The-Shelf
DAR	Data at rest
DH	Diffie-Hellman
DIT	Data in transit
DIU	Data in use
DN	Distinguished Name
EAL	Evaluation Assurance Level
ECDSA	Elliptic Curve Digital Signature Algorithm (cryptography)
ESP	Encapsulating Security Payload
EST	Enrollment over Secure Transport
GCM	Galois/Counter Mode (cryptography)
HMAC	Hashed (Hash-based) Message Authentication Code
IKE	Internet Key Exchange
IKEv2	Internet Key Exchange version 2
IPsec	Internet Protocol security (protocols)
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISA	Instruction Set Architecture
IT	Information Technology
I&A	Identification and Authentication
libest	library offering EST client and server functions

Table 20 – Terms	
Term	Definition
libcharon	library of plugins for strongSwan
libstrongswan	foundation library of the IKEv2 keying daemon
MIPS	Microprocessor without Interlocked Pipelined Stages (ISA)
MVC	MontaVista VPN Client
netlink	Linux kernel interface used for communication between kernel and user space processes and between user space processes
NIST	National Institute of Standards and Technology (USA)
OS	Operating System
PKCS #11	Public-Key Cryptography Standards API to create and manipulate cryptographic tokens
platform	Hardware, firmware, operating system, and utilities that provide the IT environment within which the TOE runs
PLT	Platform components upon which the TOE depends
PowerPC	Performance Optimization With Enhanced RISC (ISA)
POSIX	Portable Operating System Interface (standard)
PP	Protection Profile
red data	data that is proprietary, sensitive, or otherwise restricted in its distribution
RFC	Request For Comments (standards)
RISC	Reduced Instruction Set Computer (ISA)
RBG	Random Bit Generator (or Generation)
RNG	Random Number Generator (or Generation)
RVM	Reference Validation Mechanism
Security administrator	a distinguished user or role acting within the IT environment, that confers the privilege to perform platform and TOE configuration operations
SA	Security Association
SAD	Security Associations Database
SAN	Subject Alternative Name (extension to X.509 specification)
SF	Security Function
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SPD	Security Policy Database
SSL	Secure Sockets Layer
ST	Security Target
strongSwan	a multiplatform IPsec implementation available under GNU GPL
subject	a process operating on behalf of a user
systemd	system and service manager for Linux operating systems
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Functions
user	Person employing the VPN service

Table 20 – Terms	
Term	Definition
VICI	Versatile IKE Configuration Interface
Virtual Private Network	a method employing encryption to provide secure access to a remote computer over the Internet (or other unsecure network)
VPN	Virtual Private Network
VPNC	Virtual Private Network Client (term used in [11])
VPN peer	a VPN gateway, a VPN client, or a network device with a compatible VPN implementation
XFRM	Transform (Transformation)
X.509	Standard defining the format of public key certificates

References

Table 21 identifies the normative Common Criteria documents relied upon in the development of this ST.

Table 21 – Common Criteria v3.1 References	
Ref #	Title
[1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, Version 3.1 Revision 5, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.
[2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1 Revision 5, CCMB-2017-04-002, April 2017.
[3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5, CCMB-2017-04-003, April 2017.
[4]	Common Criteria for Information Technology Security Evaluation Evaluation Methodology, Version 3.1 Revision 5, CCMB-2017-04-004, April 2017.

Table 22 identifies protection profiles, guidance documents and mandatory technical supporting documents relied upon in the development of this ST.

Table 22 – PPs and Supporting Documents	
Ref #	Title
[6]	Collaborative Protection Profile for Network Devices (cPPND), Version 2.2e, March 23, 2020.

Table 23 identifies informative references consulted in the development of this ST.

Table 23 – Informative References	
Ref #	Title
[11]	VS-Anforderungsprofil VPN-Client (VPNC), Version 2.0, BSI, September 19, 2018.
[12]	CSEC Swedish Certification Body for IT Security, 188 Scheme Crypto Policy, Issue 10.0, November 03, 2020.
[r1]	https://tools.ietf.org/html/rfc4303 - IP Encapsulating Security Payload (ESP)
[r2]	https://tools.ietf.org/html/rfc7296 - Internet Key Exchange Protocol Version 2 (IKEv2)
[r3]	https://tools.ietf.org/html/rfc4945 - PKI Profile for IKE/ISAKMP/PKIX
[r4]	https://tools.ietf.org/html/rfc3602 - The AES-CBC Cipher Algorithm and Its Use with Ipsec
[r5]	https://tools.ietf.org/html/rfc4106 - The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
[r6]	https://tools.ietf.org/html/rfc4868 - Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with Ipsec

Table 23 – Informative References	
Ref #	Title
[r7]	https://tools.ietf.org/html/rfc5282 - Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol
[r8]	https://tools.ietf.org/html/rfc3526 - More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)
[r9]	https://tools.ietf.org/html/rfc5114 - Additional Diffie-Hellman Groups for Use with IETF Standards
[r10]	https://tools.ietf.org/html/rfc6954 - Using the Elliptic Curve Cryptography (ECC) Brainpool Curves for the Internet Key Exchange Protocol Version 2 (IKEv2)
[s1]	https://www.strongswan.org/apidoc/md_src_libcharon_plugins_vici_README.html - The Versatile IKE Control Interface (VICI) protocol
[s2]	https://www.opendnssec.org/softhsm/ - SoftHSM cryptographic store accessible through a PKCS #11 interface
[s3]	https://wiki.strongswan.org/projects/strongswan/wiki/SmartCards - strongSwan smart card configuration HOWTO
[s4]	https://wiki.strongswan.org/projects/strongswan/wiki/PKCS11Plugin - Using smart cards
[s5]	https://wiki.strongswan.org/projects/strongswan/wiki/LoggerConfiguration - strongSwan Logger Configuration
[s6]	https://wiki.strongswan.org/projects/strongswan/wiki/IKEv2CipherSuites - IKEv2 Cipher Suites
[s7]	https://wiki.strongswan.org/projects/1/wiki/CryptoTest - Crypto Tests
[s8]	https://wiki.strongswan.org/projects/strongswan/wiki/PluginList - strongSwan plugins
[s9]	https://wiki.strongswan.org/projects/strongswan/wiki/Charon-systemd - charon-systemd