Security Target

Ad Noctem Connect 2.3

ST Version: 1.1

April 15, 2026

Prepared for:

https://www.northropgrumman.com/

Prepared by:

TERON LABS

www.teronlabs.com

Revision History

Version	Date	Author(s)	Description of Change
1.01	25 February 2026	Teron Labs	Initial Public Release
1.1	15 April 2026	Teron Labs	Updates in response to NIAP AAR comments

Contents

List of Tables

1 Security Target Introduction

This section is the Security Target introduction. It describes the Target of Evaluation (TOE) in a narrative way at three levels of abstraction: TOE Reference, TOE Overview and TOE Description. The objective is to assist the reader in understanding the TOE and in determining that the TOE is suitable for the intended use.

The target audience is the users and the potential users of the TOE wishing to gain a precise understanding of the TOE and the security features provided. The readers are assumed to possess a good understanding of the computer and network security protocols, terms and practices. The readers are also assumed to possess some familiarity with the computer security products of Northrop Grumman Corporation.

The Security Target (ST) Introduction commences with the statements of the Security Target Reference and the TOE Reference in Sect. 1.1 and Sect. 1.2, respectively. The statement of the references is followed by the TOE Overview in Sect. 1.3 and TOE Description in Sect. 1.4

The TOE and the ST claim conformance to Common Criteria CCv3.1 Revision 5. The TOE claims conformance to the Protection Profile and a Protection Profile Module in accordance with a Protection Profile Configuration as identified in Table 1.

Table 1 TOE Conformance

Base-PP	Protection Profile for Application Software: v1.4, 7 October 2021 (PP_APP_V1.4)
PP-Module	Protection Profile Module for VPN Client: v2.4, 31 March, 2022 (MOD_VPNC_V2.4)
PP-Configuration	PP-Configuration for Application Software and Virtual Private Network (VPN) Clients, Version 1.3, 2023-04-07 (CFG_APP-VPNC_V1.3)

1.1 Security Target Reference

ST Title	Security Target Ad Noctem Connect 2.3
ST Version	1.1
ST Date:	April 15, 2026

1.2 TOE Reference

TOE Identification	Ad Noctem Connect 2.3
TOE Developer	Northrop Grumman Corporation
Evaluation Sponsor	Northrop Grumman Corporation

1.3 TOE Overview

The statement of the TOE Overview shall commence with a description of the usage and major security features of the TOE. That is followed by the statement of the TOE type. The TOE Overview shall complete with an identification of the non-TOE hardware, software and firmware required by the TOE.

1.3.1 Usage and Major Security Functions of the TOE

The TOE is the Ad Noctem Connect 2.3 by Northrop Grumman Corporation. It is an application software which implements baseline security for a software security application. The TOE is executed on a dedicated hardware platform, the GB-100 device. The TOE is not intended for execution on a general software execution platform.

The primary function of the TOE is to implement a trusted application for a VPN Client for a connection between itself and a VPN Gateway. The VPN Gateway resides at the perimeter of a private network and is controlled by the operator of the private network. The user of the TOE first connects a laptop or a PC to the platform of the TOE. The user then establishes a secure connection between the TOE and the VPN Gateway and uses that connection to communicate with the resources of a private network over an untrusted network. The secure connection ensures that any communication between the user and the resource of the private network is secure against an eavesdropper with access to the untrusted network and ability to listen to the network traffic. This is illustrated in Figure 1.

The VPN connection is implemented by the TOE acting as an IPsec peer for a VPN Server. The TOE requests a secure connection between itself and a remote VPN Server. Key exchange for IPsec is implemented with Internet Key Exchange (IKE) Version 2. Stored within the TOE are connection templates according to which the IPsec and IKEv2 protocols are executed.

The TOE is delivered installed on an appliance. The user connects the appliance physically to a PC or a Laptop computer through the ports of the appliance. Wireless connection to the appliance is not allowed. The TOE resides in the memories of the appliance and is executed in the program execution environment of the platform.

Along with the TOE, a set of templates for a secure connection are stored in the memories of the appliance. The user has no authorization to add or modify the templates but may select an instance from the set of templates stored alongside the TOE. Each template describes an IPsec profile. The VPN connection between the TOE and the VPN Server is established as guided by the template.

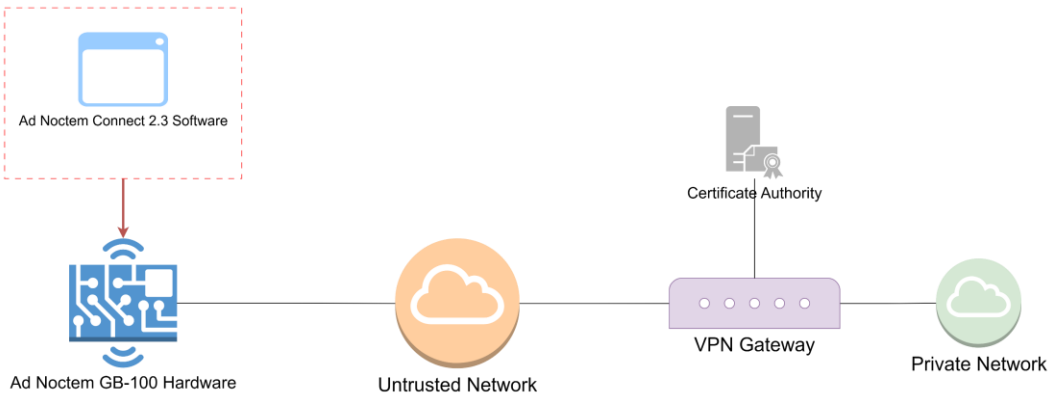


Figure 1: TOE Overview

The TOE implements only minimal interaction with the user. The user may select a template, and the TOE establishes a VPN connection between itself and the VPN gateway using the parameters stated in the template. If the connection is allowed by the VPN Gateway, the connection is established. The VPN Gateway may negotiate the connection and override the connection parameters of the template.

The appliance on which the TOE is delivered is a physical device. It is provisioned with templates, pre-shared cryptographic keys and X.509v3 public key certificates. The appliance is then issued to the end user for use, for

example, when traveling to a remote site. There are no administrative functions accessible when operational. For any administration, the TOE shall be returned to the provisioning. There is also no user authentication. The TOE only establishes the VPN connection between itself and the VPN Gateway. Any identification and authentication of the human user is by the applications of the private network.

The identities of the IPsec peers may be established with public key cryptography or pre-shared keys. When using public keys for the authentication of the IPsec peers, the cryptographic identities of the IKEv2 peers are verified using a Certification Authority (CA) controlled by the operator of the VPN Gateway. Pre-shared keys may be stored on the memories of the execution platform of the TOE at the provisioning. They may not be modified while the TOE is being used.

1.3.2 TOE Type

The TOE is an application software implementing the baseline security of an application. The TOE is not intended for a general-purpose execution environment but is only installed and executed on a dedicated platform and distributed as an appliance. The TOE implements a VPN Client which establishes an IPsec connection between itself, and a VPN Gateway as guided by the template stored on the appliance. The TOE use case resembles Use Case 3 (Communication) described in Sect. 1.4 of the Base-PP.

1.3.3 Non-TOE HSW, SW and FW required by the TOE

The TOE is an application software with the associated IPsec and IKEv2 profiles. As such, the TOE requires an execution platform and other components for being operational and to be administered. The components required by the TOE are identified and described in Table 2.

Table 2 Components Required by the TOE

Component	Description
Execution platform	The TOE is delivered pre-installed on the execution platform. The platform implements the physical connectivity to the TOE and the full software execution environment. The execution platform also implements the memories in which the TOE and the IPsec templates are stored. The TOE may only be stored and executed on the Northrop Grumman Corporation GB-100 execution platform. The execution platform is the physical appliance with the required network and user PC or laptop connectivity, memories, the entire program execution infrastructure, and the SUSE Enterprise Linux Micro OS 5.3 (Linux Kernel 5.14.21).
VPN Gateway	The TOE connects to a VPN gateway with IKEv2/IPsec support. The gateway should be operated by the operator of the Private network. It shall be configured to support the IPsec parameters required by the TOE.
Certificate Authority (CA)	The CA provides the TOE with X.509 certificates used for the authentication of the IPsec connection endpoints. The CA is operated by the operator of the VPN Gateway.
User PC or Laptop	The user of the TOE connects the PC or Laptop physically to the appliance on which the TOE resides and is executed. The TOE establishes a VPN connection between itself and the VPN Gateway and uses the VPN connection for securing the communication between the TOE and the VPN Gateway. The applications executed on the User PC or Laptop use the VPN connection for secure access to the resources of the Private Network.

User Connectivity	The User PC or Laptop must be connected physically to the appliance on which the TOE resides and is executed. Wireless connection must not be used. Physical connection must be done using a network cable. The network cable must be provided by the organization of the user. Unknown cables must not be used for connecting the User PC or Laptop to the TOE.
Provisioning Environment	The TOE is provisioned in a secure environment by the organizational administrator. Once provisioned, the TOE is issued to the end user for use at remote sites. All administration of the TOE takes place in the provisioning environment by the organizational administrator. The organizational administrator uses the interfaces of the TOE platform to modify the TOE configuration files stored on the platform memories.
Private Network	The Private Network is a secure network operated by the organization of the user. In the Private Network the organization implements services to which the user connects over an insecure network using the User PC or Laptop. The user executes programs or accesses services in the Private Network over the VPN connection between the TOE and the VPN Gateway.

1.4 TOE Description

This section describes the physical scope of the TOE and the logical scope of the TOE.

1.4.1 Physical Scope of the TOE

The TOE does not include any hardware even if the TOE may only be executed on a dedicated platform. The physical scope of the TOE includes the TOE Software, IPsec templates, and the security guidance. The components constituting the physical scope of the TOE are identified and described in Table 3.

Table 3 Physical Scope of the TOE

Component	Description
TOE Software	TOE Software implements all the functions of the TOE. The TOE Software is only executed on a dedicated execution platform. The execution platform boots up the TOE when the appliance is powered on. The boot process includes verification of the integrity of the critical cryptographic and other libraries used by the TOE. If any of the verification fails, the platform shall not boot up the TOE. The TOE Software is identified as Ad Noctem Connect 2.3. The TOE software image is version v2.3. Linked to the TOE software is the OpenSSL 1.1.1 FIPS module which implements each cryptographic function of the TOE. The TOE is an application and kernel module running on a Linux operating system as illustrated in Figure 1 in Sect. 1.3 of the Base-PP. Specifically, the IPsec functionality is implemented in the kernel module. The TOE use case resembles the Use Case 3 (Communication) described in Sect. 1.4 of the Base-PP. The communication may be any communication between the applications running on the User PC or Laptop and the corresponding application or service in the Private Network.
IPsec Templates	Along with the TOE Software are the IPsec templates stored in the memories of the platform. The organizational administrator may define additional templates and remove or modify existing templates.

	Alongside the TOE Software, three pre-configured templates are delivered installed on the memories of the execution platform: – adnoctem-cert.template – adnoctem-psk-roaming.template – adnoctem-psk.template
TOE Security Guidance	TOE Security Guidance is the guidance supplement which enhances the product security guidance and instructs the users and administrators of the TOE in the secure use of the TOE. The security guidance is downloaded from the developer's web site in PDF format. The TOE Security Guidance is Ad Noctem Connect 2.3 on GB-100 Platform Common Criteria Security Guidance Supplement v1.0.

1.4.2 Logical Scope of the TOE

The logical scope of the TOE is constituted of the security function the TOE implements. The logical scope of the TOE is described in Table 4.

Table 4 Logical Scope of the TOE

Security Function	Description
Cryptographic Support	The TOE incorporates OpenSSL 1.1.1 FIPS module and the Linux Kernel Crypto architecture for cryptographic support. The cryptographic architecture is used for implementing IPsec and IKEv2 for the VPN Client functionality, and for the internal storage of cryptographic keys and other critical security parameters stored persistently on the TOE. Each cryptographic algorithm implemented by the TOE has been validated through the Cryptographic Algorithm Validation Program (CAVP) testing. Only CAVP validated cryptographic implementations are used by the TOE.
User Data Protection and Privacy	The TOE does not collect any sensitive user data. It protects user data that flows through it, as it is configured. The TOE does not store or transmit over a network any Personally Identifiable Information (PII). The TOE is not a general-purpose application, and the execution platform does not offer any general computing facilities. Therefore, the TOE does not have access to any general information repositories which might contain sensitive information about the user. The TOE only uses the network connectivity and the connectivity of the end user PC or laptop as provided by the platform.
Peer Entity Authentication	The TOE authenticates the IKEv2 peer entities when establishing the IPsec connection. The peer entity of the TOE is the VPN Gateway. Authentication of the peer entity is by cryptographic means using pre-shared keys or X.509 certificates. If the cryptographic authentication or the certificate validation fails, the TOE shall reject the connection. User data shall not be transmitted if the VPN Connection is not successfully established.
Protection	The TOE is implemented in accordance with good security engineering practices. Only trusted third party libraries are used, and the security guidance of the platform developer is followed. The TOE is of a known version, and the version numbering is used consistently. Prior to the execution of the TOE, the platform

	executes a suite of tests to verify the integrity of the kernel and the cryptographic libraries used by the TOE. The TOE shall only be executed only if the self-tests pass.
Trusted Paths and Channels	The TOE implements an IPsec tunnel for the protection of the user data communicated between the TOE and the associated VPN Gateway. All communication between the applications running on the User PC or Laptop and the resources of the Private Network take place over the IPsec tunnel.

2 Conformance Claims

This section states the Conformance Claims for the ST and the TOE. This includes a statement of the Conformance Claims, a statement of the Conformance Claim Rationale, and the Identification of the Technical Decisions applicable to the TOE.

2.1 Statement of Conformance Claims

The ST and the TOE claim conformance to Common Criteria Version 3.1 Revision 5, Part 1 through to Part 3 identified in the following:

- Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, April 2017, Version 3.1 Revision 5, CCMB-2017-04-001
- Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-002
- Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components, April 2017, Version 3.1 Revision 5, CCMB-2017-04-003

The ST claims CC Part 2 conformance as CC Part 2 Extended.

The ST claims CC Part 3 conformance as CC Part 3 Extended.

The ST claims conformance to the following Protection Profile, and the Protection Profile Module:

- Protection Profile for Application Software: v1.4, 7 October 2021 (PP_APP_V1.4)
- Protection Profile Module for VPN Client: v2.4, 31 March 2022 (MOD_VPNC_V2.4)

Conformance to the Base-PP and the PP-Module is claimed in accordance with the PP-Configuration:

- PP-Configuration for Application Software and Virtual Private Network (VPN) Clients, Version 1.3, 2023-04-07 (CFG_APP-VPNC_V1.3).

The ST claims no conformance to any Evaluation Assurance Level or any other security assurance requirement package. Security assurance requirements applicable to the TOE are those drawn from the Base-PP as required by CFG_APP-VPNC_V1.3.

The ST claims conformance to Protection Profile for Application Software: v1.4, Date: 7-October-2021 (PP_APP_V1.4) as PP-conformant.

The ST claims conformance to the PP-Configuration for Application Software and Virtual Private Network (VPN) Clients, Version: 1.3, 2023-04-07 (CFG_APP-VPNC_V1.3) as PP-configuration-conformant.

The ST claims exact conformance to the Base-PP, exact conformance to each PP-Module, and exact conformance to the PP-configuration. Exact conformance is defined in CC and CEM addenda for Exact Conformance, Selection-Based SFRs, and Optional SFRs, CCDB-013-v2.0 Final, 2021-Sep-30.

2.2 Conformance Claim Rationale

This section states the Conformance Claim rationale to justify the conformance claims.

2.2.1 TOE Type Consistency Rationale

The TOE is a software application that implements VPN functionality. It implements a set of security features required for exact conformance with the Base-PP and with the PP-Module. The PP and the PP-Module are used in accordance with the PP-Configuration. These are exactly the PP, the PP-Module, and the PP-configuration claimed

in Sect. 2.1. The PP and the PP-Modules are exactly as identified in Sect. 1.3 of the PP-Configuration. This ensures that the TOE Type is consistent with the TOE Type in the Base-PP, PP-Modules, and PP-Configuration.

2.2.2 Security Problem Definition Consistency

The statement of the Security Problem Definition in this ST is reproduced exactly from the Base-PP and from the claimed PP-Modules. The resulting Security Problem Definition is a union of the Security Problem Definition of the Base-PP and the PP-Modules. There are no additional Security Problem Definition elements included in the statement of the Security Problem Definition. This ensures that the statement of the Security Problem Definition is consistent with the PP-Configuration.

2.2.3 Security Objective Consistency

The statement of the Security Objectives in this ST is reproduced exactly from the Base-PP and the PP-Modules. The resulting Security Objectives statement is a union of the Security Objectives of the Base-PP and the PP-Modules. There are no additional Security Objectives included in the statement of the Security Objectives. This ensures that the statement of the Security Objectives is consistent with the PP-Configuration.

2.2.4 Security Requirements Consistency

The security functional requirements are drawn exactly from the Base-PP and the PP-Module. The statement of the security functional requirements includes all mandatory security requirements and those selection-based security functional requirements applicable to the TOE. The developer claims no optional requirements and does not include additional components in the statement of the security functional requirements. As such, the security functional requirements are consistently drawn from the Base-PP and the PP-Modules, and the ST ensures the consistency of the security functional requirements.

The security assurance requirements are drawn from the Base-PP only. This is consistent with Sect. 2.2 of the PP-Configuration. This ensures the consistency of the security assurance requirements.

2.3 Technical Decisions

The Technical Decisions (TD) applicable to the Base-PP and the PP-Module are given. Each TD is identified and described. The applicability of the TD is stated and, if the TD is not applicable to the TOE, a brief rationale for the exclusion of the TD is given. There are no Technical Decisions appliable to the PP-Configuration.

2.3.1 Technical Decisions Applicable to the Application Software Version 1.4

The Technical Decisions applicable to the Base-PP are given in Table 5.

Table 5 Technical Decisions applicable to the Base-PP

TD	Description	Applicable	Exclusion Rationale (if applicable)
TD0964	Clarifications to FMT_MEC_EXT.1 Windows Test	Yes	The TOE does not execute on Microsoft Windows, this TD is applicable to the test EA, but not the TOE.
TD0945	Updating FIPS 186-4 to 186-5 in PP_APP_V1.4	No	FCS_CKM.1/AK is claimed through MOD_VPNC_v2.4, not the Base-PP. Therefore, this TD from the Base-PP is not applicable
TD0931	Clarification when CTR_DRBG is Selected for FCS_RBG_EXT.2.2 in PP_APP_V1.4	Yes	

TD0914	Addition of PKG_TLS_V2.0 to Conformance Claims	Yes	Applicable to the Base-PP, however the TOE does not claim the functional package for TLS.
TD0865	Consistency of Cryptographic Key Sizes	Yes	
TD0844	Addition of Assurance Package for Flaw Remediation V1.0 Conformance Claim	Yes	
TD0823	Update to Microsoft Windows Exploit Protection link in FPT_AEX_EXT.1.3	Yes	The TOE does not execute on Microsoft Windows, this TD is applicable to the test EA, but not the TOE.
TD0822	Correction to Windows Manifest File for FDP_DEC_EXT.1	Yes	The TOE does not execute on Microsoft Windows.
TD0815	Addition of Conditional TSS Activity for FPT_AEX_EXT.1.5	Yes	
TD0798	Static Memory Mapping Exceptions	Yes	
TD0780	FIA_X509_EXT.1 Test 4 Clarification	Yes	
TD0756	Update for platform-provided full disk encryption	Yes	
TD0747	Configuration Storage Option for Android	Yes	The TOE does not execute on Android, this TD is applicable to the test EA, but not the TOE.
TD0743	FTP_DIT_EXT.1.1 Selection exclusivity	No	FPT_DIT_EXT.1 is claimed through MOD_VPNC_2.4, as such, this PP_APP TD is not applicable.
TD0736	Number of elements for iterations of FCS_HTTPS_EXT.1	No	HTTPS is not claimed by the TOE.
TD0719	ECD for PP APP V1.3 and 1.4	Yes	
TD0717	Format changes for PP_APP_V1.4	Yes	FCS_CKM.1/AK and FCS_COP.1/SKC are not affected by this TD as they are taken from MOD_VPNC_2.4.
TD0664	Testing activity for FPT_TUD_EXT.2.2	No	FPT_TUD_EXT.2 is not claimed by the TOE.
TD0650	Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4	Yes	
TD0628	Addition of Container Image to Package Format	No	FPT_TUD_EXT.2 is not claimed by the TOE.

2.3.2 Technical Decisions Applicable to the PP-Module for VPN Client 2.4

The Technical Decisions applicable to the PP-Module are given in Table 6.

Table 6 Technical Decisions Applicable to PP-Module

TD	Description	Applicable	Exclusion Rationale (if applicable)
TD0897	RFC 8784 Optional in VPNC 2.4	Yes	
TD0876	Updating FIPS 186-4 to 186-5 in MOD_VPNC_V2.4	Yes	
TD0824	Aligning MOD_VPNGW 1.3 with NDcPP 3.0E	Yes	
TD0788	Terminology Change in MOD_VPNC: Extended to Functional Package	Yes	
TD0753	MOD_VPNC FTP_DIT_EXT.1 Alignment for App PP 1.4	Yes	
TD0725	Correction to FCS_CKM_EXT.2/4 selections	Yes	
TD0711	FMT_SMF.1 direction when using MDF 3.3	No	Conformance with MDF PP is not being claimed in the ST.
TD0690	Missing EAs for FDP_VPN_EXT.1	No	The TOE does not claim FDP_VPN_EXT.1, as the MDF PP is not claimed in the ST.
TD0672	VPN Client PP-Module updated to allow for new PP and PP-Module Versions	Yes	
TD0662	Changes to Testing IPsec NAT Transversal and XAUTH in MOD_VPNC 2.4	Yes	
TD0647	Table 2 Applicability	No	TOE does not implement audit log functions.

3 Security Problem Definition

The Security Problem Definition includes a statement of the Threats, Assumptions and OSPs applicable to the TOE. Each is stated in this section.

3.1 Threats

The threats applicable to the TOE are drawn from the Base-PP and from the PP-Module. The threats drawn from the Base-PP as applicable are given in Table 7. The threats drawn from the PP-Module are given in Table 8. There are no additions or omissions, and the wording of each threat statement is taken verbatim.

Table 7 Threats Drawn from the Base-PP

Threat ID	Threat Statement
T.NETWORK_ATTACK	An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with the application software or alter communications between the application software and other endpoints in order to compromise it.
T.NETWORK_EAVESDROP	An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between the application and other endpoints.
T.LOCAL_ATTACK	An attacker can act through unprivileged software on the same computing platform on which the application executes. Attackers may provide maliciously formatted input to the application in the form of files or other local communications.
T.PHYSICAL_ACCESS	An attacker may try to access sensitive data at rest.

Table 8 Threats Drawn from the PP-Module

Threat ID	Threat Statement
T.UNAUTHORIZED_ACCESS	This PP-Module does not include requirements that can protect against an insider threat. Authorized users are not considered hostile or malicious and are trusted to follow appropriate guidance. Only authorized personnel should have access to the system or device that contains the IPsec VPN client. Therefore, the primary threat agents are the unauthorized entities that try to gain access to the protected network (in cases where tunnel mode is used) or to plaintext data that traverses the public network (regardless of whether transport mode or tunnel mode is used). The endpoint of the network communication can be both geographically and logically distant from the TOE and can pass through a variety of other systems. These intermediate systems may be under the control of the adversary and offer an opportunity for communications over the network to be compromised. Plaintext communication over the network may allow critical data (such as passwords, configuration settings, and user data) to be read or manipulated directly by a malicious user or process on intermediate systems, leading to a compromise of the TOE or to the secured environmental systems that the TOE is being used to

	facilitate communications with. IPsec can be used to provide protection for this communication; however, there are numerous options that can be implemented for the protocol to be compliant to the protocol specification listed in the RFC. Some of these options can have negative impacts on the security of the connection. For instance, using a weak encryption algorithm (even one that is allowed by the RFC, such as DES) can allow an adversary to read and even manipulate the data on the encrypted channel, thus circumventing countermeasures in place to prevent such attacks. Further, if the protocol is implemented with little-used or non-standard options, it may be compliant with the protocol specification but will not be able to interact with other diverse equipment that is typically found in large enterprises. Even though the communication path is protected, there is a possibility that the IPsec peer could be tricked into thinking that a malicious third-party user or system is the TOE. For instance, a middleman could intercept a connection request to the TOE and respond to the request as if it were the TOE. In a similar manner, the TOE could also be tricked into thinking that it is establishing communications with a legitimate IPsec peer when in fact it is not. An attacker could also mount a malicious man-in-the-middle-type of attack, in which an intermediate system is compromised, and the traffic is proxied, examined, and modified by this system. This attack can even be mounted via encrypted communication channels if appropriate countermeasures are not applied. These attacks are, in part, enabled by a malicious attacker capturing network traffic (for instance, an authentication session) and "playing back" that traffic in order to fool an endpoint into thinking it was communicating with a legitimate remote entity.
T.TSF_CONFIGURATION	Configuring VPN tunnels is a complex and time-consuming process, and prone to errors if the interface for doing so is not well-specified or well-behaved. The inability or failure of an ignorant or careless administrator to configure certain aspects of the interface may also lead to the misspecification of the desired communications policy or use of cryptography that may be desired or required for a particular site. This may result in unintended weak or plaintext communications while the user thinks that their data are being protected. Other aspects of configuring the TOE or using its security mechanisms (for example, the update process) may also result in a reduction in the trustworthiness of the VPN client.
T.USER_DATA_REUSE	Data traversing the TOE could inadvertently be sent to a different user as a consequence of a poorly-designed TOE; since these data may be sensitive, this may cause a compromise that is unacceptable. The specific threat that must be addressed concerns user data that is retained by the TOE in the course of processing network traffic that could be inadvertently re-used in sending network traffic to a user other than that intended by the sender of the original network traffic.
T.TSF_FAILURE	Security mechanisms of the TOE generally build up from a primitive set of mechanisms (e.g., memory management, privileged modes of process execution) to more complex sets of mechanisms. Failure of the primitive mechanisms could lead to a compromise in more complex mechanisms, resulting in a compromise of the TSF.

3.2 Assumptions

The assumptions applicable to the TOE are drawn from the Base-PP and the applicable PP-Modules. There are no additions or omissions, and the wording of each assumption statement is taken verbatim. The assumptions drawn from the Base-PP are given in Table 9. The assumptions drawn from the PP-Module are given in Table 10.

Table 9 Assumptions Drawn from the Base-PP

Assumption ID	Assumption Statement
A.PLATFORM	The TOE relies upon a trustworthy computing platform with a reliable time clock for its execution. This includes the underlying platform and whatever runtime environment it provides to the TOE
A.PROPER_USER	The user of the application software is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy.
A.PROPER_ADMIN	The administrator of the application software is not careless, willfully negligent or hostile, and administers the software in compliance with the applied enterprise security policy.

Table 10 Assumptions Drawn from the PP-Module

Assumption ID	Assumption Statement
A.NO_TOE_BYPASS	Information cannot flow onto the network to which the VPN client's host is connected without passing through the TOE.
A.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.
A.TRUSTED_CONFIG	Personnel configuring the TOE and its OE will follow the applicable security configuration guidance.

3.3 Organizational Security Policies

Neither the Base-PP nor the PP-Module defines Organizational Security Policies. Therefore, there are none stated in the ST.

4 Security Objectives

The security objectives are stated for the TOE in Sect. 4.1 and for the operational environment of the TOE in Sect. 4.2. The security objectives rationale is given in Sect. 4.3.

4.1 Security Objectives for the TOE

The security objectives for the TOE are drawn verbatim from the Base-PP. They are stated in Table 11.

The security objectives drawn from the PP-Module are reproduced in part. The statement does not include the paragraph from the PP Module which references the SFRs that address the corresponding security objectives. The security objectives for the TOE drawn from the PP-Module are given in Table 12.

Table 11 Security Objectives for the TOE Drawn from the Base-PP

Security Objective ID	Security Objective Statement
O.INTEGRITY	Conformant TOEs ensure the integrity of their installation and update packages, and also leverage execution environment-based mitigations. Software is seldom, if ever, shipped without errors. The ability to deploy patches and updates to fielded software with integrity is critical to enterprise network security. Processor manufacturers, compiler developers, execution environment vendors, and operating system vendors have developed execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems. Application software can often take advantage of these mechanisms by using APIs provided by the runtime environment or by enabling the mechanism through compiler or linker options.
O.QUALITY	To ensure quality of implementation, conformant TOEs leverage services and APIs provided by the runtime environment rather than implementing their own versions of these services and APIs. This is especially important for cryptographic services and other complex operations such as file and media parsing. Leveraging this platform behavior relies upon using only documented and supported APIs.
O.MANAGEMENT	To facilitate management by users and the enterprise, conformant TOEs provide consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the deployment of applications and application updates through the use of platform-supported deployment mechanisms and formats, as well as providing mechanisms for configuration. This also includes providing control to the user regarding disclosure of any PII.
O.PROTECTED_STORAGE	To address the issue of loss of confidentiality of user data in the event of loss of physical control of the storage medium, conformant TOEs will use data-at-rest protection. This involves encrypting data and keys stored by the TOE in order to prevent unauthorized access to this data. This also includes unnecessary network communications whose consequence may be the loss of data.
O.PROTECTED_COMMS	To address both passive (eavesdropping) and active (packet modification) network attack threats, conformant TOEs will use a trusted channel for sensitive data. Sensitive data includes cryptographic keys, passwords, and any other data specific to the application that should not be exposed outside of the application.

Table 12 Security Objectives for the TOE Drawn from the PP-Module

Security Objective ID	Security Objective Statement
O.AUTHENTICATION	To address the issues associated with unauthorized disclosure of information in transit, a compliant TOE's authentication ability (IPsec) will allow the TSF to establish VPN connectivity with a remote VPN gateway or peer and ensure that any such connection attempt is both authenticated and authorized.
O.CRYPTOGRAPHIC_FUNCTIONS	To address the issues associated with unauthorized disclosure of information in transit, a compliant TOE will implement cryptographic capabilities. These capabilities are intended to maintain confidentiality and allow for detection and modification of data that is transmitted outside of the TOE.
O.KNOWN_STATE	The TOE will provide sufficient measures to ensure it is operating in a known state. At minimum this includes management functionality to allow the security functionality to be configured and self-test functionality that allows it to assert its own integrity. It may also include auditing functionality that can be used to determine the operational behavior of the TOE.
O.NONDISCLOSURE	To address the issues associated with unauthorized disclosure of information at rest, a compliant TOE will ensure that non-persistent data is purged when no longer needed. The TSF may also implement measures to protect against the disclosure of stored cryptographic keys and data through implementation of protected storage and secure erasure methods. The TOE may optionally also enforce split-tunnelling prevention to ensure that data in transit cannot be disclosed inadvertently outside of the IPsec tunnel and prohibit transmission of packets through a connection until certain conditions are met.

4.2 Security Objectives for the Operational Environment

The security objectives for the operational environment are drawn from the Base-PP and the PP-Module. The security objectives for the operational environment are drawn in verbatim from the Base-PP and are stated in Table 13. The security objectives for the environment drawn from the PP-Module are given in Table 14.

Table 13 Security Objectives for the Operational Environment Drawn from the Base-PP

Security Objective ID	Security Objective Statement
OE.PLATFORM	The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the TOE.
OE.PROPER_USE	The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.
OE.PROPER_ADMIN	The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

Table 14 Security Objectives for the Operational Environment Drawn from the PP-Module

Security Objective ID	Security Objective Statement
OE.NO_TOE_BYPASS	Information cannot flow onto the network to which the VPN client's host is connected without passing through the TOE.
OE.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the environment.
OE.TRUSTED_CONFIG	Personnel configuring the TOE and its OE will follow the applicable security configuration guidance.

4.3 Security Objectives Rationale

The statement of the security problem definition and the statements of the security objectives are drawn verbatim from the Base-PP and the PP-Module. Therefore, the security objectives rationales given in Sect. 4.3 of the Base-PP and within the statement of each security objective in the PP-Module are directly applicable to the ST. They are not repeated here.

5 Security Requirements

This section states the security requirements applicable to the TOE. The statement commences with the extended components definition in Sect. 5.1. The statement of the extended components is followed by the statement of the notations and conventions used in the expression of the security requirements. The security functional requirements are summarized in Sect. 5.3 and stated in the subsequent subsections on a per source and functional class basis. The security assurance requirements are only drawn from the Base-PP and are given in Sect. 5.6. The security requirements rationale is given in Sect. 5.7.

5.1 Extended Components Definition

The ST references several extended components. Each one is taken verbatim from the Base-PP or the PP-Modules. Only the operations allowed in the statement of the extended components are implemented in the ST. There are no additional or modified extended components included in the ST. Therefore, the statement of the extended components is exactly as in the Base-PP and the PP-Modules. They are not repeated here.

5.2 Notation and Conventions

This ST follows the specific conventions in the completion of the operations on the Security Functional Requirements. The following conventions are followed to indicate the operations:

- Unaltered Security Functional Requirements are stated using the notation given in CC Part 2 or in the applicable extended component definition.
- When a refinement made in the ST, the added text is indicated with a **bold font,** and any removal of text is indicated with a ~~strikethrough~~.
- When a selection is completed in the ST, the selected values are indicated with underlined text.
    - For example, a selection "[selection: disclosure, modification, loss of use]" in a Security Functional Requirement drawn from the Base-PP or PP-Module might become "[disclosure]" when the selection is performed in the ST.
- Assignment completed in the ST is indicated with *italicized font*.
- Assignment completed within a selection in the ST is indicated with *italicized and underlined font*.
    - For example, an assignment within a selection "[selection: change_default, query, modify, delete, [assignment: other operations]]" in a Security Functional Requirement drawn from the Base-PP or PP-Module might become "[change_default,[*select_tag*]]" when both the selection and the assignment are completed in the ST.
- Iteration is indicated by adding a descriptive string starting with "/" (e.g. "FCS_COP.1/Hash").
- Extended requirements are indicated using the notation given in the Base-PP or PP-Module from which they are drawn. Each extended Security Functional Requirement is indicated with a label "_EXT" at the end of the requirement name (e.g. FCS_RBG_EXT).

When the Base-PP or a PP-Module uses an alternative notation or expression for the statement of a Security Functional Requirements, that notation or expression is followed in the ST - possibly with the addition of the above conventions. This includes, for example,

- The capitalization of the component names is followed in verbatim even if sometimes inconsistent, and
- The PP-Module alternatives for selection operations are given in italic font. The italic font is maintained and additionally also underlined to indicate that the selection is performed from the set of allowed values.

The Security Assurance Requirements are drawn from the Base-PP only for conformance with the PP-Configuration. There are no operations defined for the Security Assurance Requirements. The notation for expressing the Security Assurance Requirements is taken verbatim from the Base-PP.

5.3 Security Functional Requirements Summary

The Security Functional Requirements applicable to the TOE as drawn from different sources are summarized in Table 15 and Table 16. On those occasions where a PP-Module refines the statement of a security functional component of a Base-PP, the component is listed under the Base-PP with an indication in the statement in the SFR of the SFR being refined as per the PP-Module. Any modification to the statement of Security Functional Requirements mandated by a Technical Decision are also indicated in the statement of the requirement.

Table 15 SFR Summary for the Base-PP

Security Functional Class	Security Functional Component
FCS: Cryptographic Support	FCS_CKM.1 Cryptographic Key Generation Services FCS_CKM.1/AK Cryptographic Asymmetric Key Generation FCS_COP.1/Hash Cryptographic Operation - Hashing FCS_COP.1/KeyedHash Cryptographic Operation - Keyed-Hash Message Authentication FCS_COP.1/Sig Cryptographic Operation – Signing FCS_COP.1/SKC Cryptographic Operation - Encryption/Decryption FCS_RBG_EXT.1 Random Bit Generation Services FCS_RBG_EXT.2 Random Bit Generation from Application FCS_STO_EXT.1 Storage of Credentials
FDP: User Data Protection	FDP_DEC_EXT.1 Access to Platform Resources FDP_NET_EXT.1 Network Communications FDP_DAR_EXT.1 Encryption Of Sensitive Application Data
FIA: Identification and Authentication	FIA_X509_EXT.1 X.509 Certificate Validation FIA_X509_EXT.2 X.509 Certificate Authentication
FMT: Security Management	FMT_MEC_EXT.1 Supported Configuration Mechanism FMT_CFG_EXT.1 Secure by Default Configuration FMT_SMF.1 Specification of Management Functions
FPR: Privacy	FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information
FPT: Protection of the TSF	FPT_API_EXT.1 Use of Supported Services and APIs FPT_AEX_EXT.1 Anti-Exploitation Capabilities FPT_IDV_EXT.1 Software Identification and Versions FPT_LIB_EXT.1 Use of Third Party Libraries

	FPT_TUD_EXT.1 Integrity for Installation and Update
FTP: Trusted Path/Channel	FTP_DIT_EXT.1 Protection of Data in Transit

Table 16 SFR Summary for the PP-Module

Security Functional Class	Security Functional Component
FCS: Cryptographic Support	FCS_CKM.1/VPN Cryptographic Key Generation Services FCS_CKM_EXT.2 Cryptographic Key Storage FCS_CKM_EXT.4 Cryptographic Key Destruction FCS_IPSEC_EXT.1 IPsec
FDP: User Data Protection	FDP_RIP.2 Full Residual Information Protection
FIA: Identification and Authentication	FIA_PSK_EXT.1 Pre-Shared Key Composition FIA_PSK_EXT.2 Generated Pre-Shared Keys
FMT: Security management	FMT_SMF.1/VPN Specification of Management Functions (VPN)
FPT: Protection of the TSF	FPT_TST_EXT.1/VPN TSF Self-Test

5.4 Security Functional Requirements Drawn from the Base-PP

5.4.1 Class FCS: Cryptographic Support

5.4.1.1 FCS_CKM.1[1] Cryptographic Key Generation Services

FCS_CKM.1.1 The application shall [*implement asymmetric key generation*].

5.4.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation

FCS_CKM.1.1/AK[2] The **application** shall [

- ***implement functionality***

] to generate **asymmetric** cryptographic keys in accordance with a specified cryptographic key generation algorithm

- ***[ECC schemes]*** using **["NIST curves" P-384, and [P-256, *P-521*]]** that meet the following: **[FIPS PUB 186-5, "Digital Signature Standard (DSS)", Appendix B.4],** and,

[

- *[FFC Schemes] using **["safe-prime" groups]** that meet the following: **[NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"** and [RFC 3526, RFC 7919]].*

].

[1] As per MOD_VPNC_V2.4
[2] As per MOD_VPNC_V2.4 and as per TD0876.

5.4.1.3 FCS_CKM.2 Cryptographic Key Establishment

FCS_CKM.2.1[3] The application shall [*implement functionality*] to perform cryptographic key establishment in accordance with a specified key establishment method: [

- *[Elliptic curve-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]*; and

[

- *[Finite field-based key establishment schemes] that meets the following: [NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]*

].

5.4.1.4 FCS_COP.1/Hash Cryptographic Operation – Hashing

FCS_COP.1/Hash[4] The application shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm [

- *SHA-256,*
- *SHA-384,*
- *SHA-512,*

] and message digest sizes [

- *256,*
- *384,*
- *512*

] bits that meet the following: [FIPS Pub 180-4].

5.4.1.5 FCS_COP.1/KeyedHash Cryptographic Operation - Keyed-Hash Message Authentication

FCS_COP.1/KeyedHash[5] The application shall perform [*keyed-hash message authentication*] in accordance with a specified cryptographic algorithm [

- HMAC-SHA-256,
- HMAC-SHA-384,
- HMAC-SHA-512

] and [

- *no other algorithms*

] with key sizes [*256, 384, 512 bits*] and message digest sizes [256, *384, 512*] and [*no other size*] bits that meet the following: [*FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code' and FIPS Pub 180-4 'Secure Hash Standard'*].

[3] As per MOD_VPNC_V2.4
[4] As per TD0717
[5] As per TD0717

5.4.1.6 FCS_COP.1/Sig Cryptographic Operation – Signing

FCS_COP.1.1/Sig[6] The application shall perform *[cryptographic signature services (generation and verification)]* in accordance with a specified cryptographic algorithm [

- *RSA schemes using cryptographic key sizes of [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5],*
- *ECDSA schemes using ["NIST curves" P-256, P-384 and [P-521]] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6]*

].

5.4.1.7 FCS_COP.1/SKC Cryptographic Operation - Encryption/Decryption

FCS_COP.1.1/SKC[7] The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm

- AES-CBC (as defined in NIST SP 800-38A) mode,
- AES-GCM (as defined in NIST SP 800-38D) mode,

and [

- *no other modes*

] and cryptographic key sizes [*128-bit, 256-bit*].

5.4.1.8 FCS_RBG_EXT.1 Random Bit Generation Services

FCS_RBG_EXT.1.1 The application shall [

- *implement DRBG functionality*

] for its cryptographic operations.

5.4.1.9 FCS_RBG_EXT.2 Random Bit Generation from Application

FCS_RBG_EXT.2.1 The application shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [*CTR_DRBG (AES)*]

FCS_RBG_EXT.2.2[8] The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a platform-based DRBG and [

- *a hardware-based noise source*

] with a minimum of [

- *256 bits,*

] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

[6] As per TD0717
[7] As per MOD_VPNC_V2.4
[8] As per TD0931

5.4.1.10 FCS_STO_EXT.1 Storage of Credentials

FCS_STO_EXT.1.1[9] The application shall [

- securely store *[X.509v3 certificates, pre-shared cryptographic keys for IKE, and connection templates]* with platform provided [
    - [
        - AES-XTS (as defined in NIST SP 800-38E) mode

    ] and cryptographic key size of 256-bits.

] to non-volatile memory.

5.4.2 Class FDP: User Data Protection

5.4.2.1 FDP_DEC_EXT.1 Access to Platform Resources

FDP_DEC_EXT.1.1 The application shall restrict its access to [

- *network connectivity*

].

FDP_DEC_EXT.1.2 The application shall restrict its access to [

- no sensitive information repositories

].

5.4.2.2 FDP_NET_EXT.1 Network Communications

FDP_NET_EXT.1.1 The application shall restrict network communication to [

- user-initiated communication for [*VPN Connection*]

].

5.4.2.3 FDP_DAR_EXT.1.1 Encryption of Sensitive Application Data

FDP_DAR_EXT.1.1 The application shall [

- leverage platform-provided functionality to encrypt sensitive data

] in non-volatile memory.

5.4.3 Class FIA: Identity and Authentication

5.4.3.1 FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.1.1 The application shall [*invoke platform-provided functionality*] to validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.

[9] As per TD0865

- The application shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The application shall validate that any CA certificate includes caSigning purpose in the key usage field
- The application shall validate the revocation status of the certificate using [*CRL as specified in RFC 8603*].
- The application shall validate the extendedKeyUsage (EKU) field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing Purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the EKU field.
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
    - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the EKU field.
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.
    - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the EKU field.

FIA_X509_EXT.1.2 The application shall treat a certificate as a CA certificate only if the basicConstraints extension is present and the CA flag is set to TRUE.

5.4.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

FIA_X509_EXT.2.1[10] The application shall use X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec and [*no other protocols*].

FIA_X509_EXT.2.2 When the application cannot establish a connection to determine the validity of a certificate, the TSF shall [*not accept the certificate*].

5.4.4 Class FMT: Security Management

5.4.4.1 FMT_MEC_EXT.1 Supported Configuration Mechanism

FMT_MEC_EXT.1.1 The application shall [*invoke the mechanisms recommended by the platform vendor for storing and setting configuration options*].

5.4.4.2 FMT_CFG_EXT.1 Secure by Default Configuration

FMT_CFG_EXT.1.1 The application shall provide only enough functionality to set new credentials when configured with default credentials or no credentials.

FMT_CFG_EXT.1.2 The application shall be configured by default with file permissions which protect the application binaries and data files from modification by normal unprivileged users.

5.4.4.3 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions [

[10] As per MOD_VPNC_V2.4, and as per TD0788.

- *no management functions*

].

5.4.5 Class FPR: Privacy

5.4.5.1 FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information

FPR_ANO_EXT.1.1 The application shall [

- *not transmit PII over a network,*

].

5.4.6 Class FPT: Protection of the TSF (FPT)

5.4.6.1 FPT_API_EXT.1 Use of Supported Services and APIs

FPT_API_EXT.1.1 The application shall use only documented platform APIs.

5.4.6.2 FPT_AEX_EXT.1 Anti-Exploitation Capabilities

FPT_AEX_EXT.1.1 The application shall not request to map memory at an explicit address except for [*no exceptions*].

FPT_AEX_EXT.1.2 The application shall [

- *not allocate any memory region with both write and execute permissions*

].

FPT_AEX_EXT.1.3 The application shall be compatible with security features provided by the platform vendor.

FPT_AEX_EXT.1.4 The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.

FPT_AEX_EXT.1.5 The application shall be built with stack-based buffer overflow protection enabled.

5.4.6.3 FPT_IDV_EXT.1 Software Identification and Versions

FPT_IDV_EXT.1.1 The application shall be versioned with [[*sequential versioning numbers*]].

5.4.6.4 FPT_LIB_EXT.1 Use of Third Party Libraries

FPT_LIB_EXT.1.1 The application shall be packaged with only [*OpenSSL, FIPSCheck,*].

5.4.6.5 FPT_TUD_EXT.1 Integrity for Installation and Update

FPT_TUD_EXT.1.1 The application shall [*leverage the platform*] to check for updates and patches to the application software.

FPT_TUD_EXT.1.2 The application shall [*leverage the platform*] to query the current version of the application software.

FPT_TUD_EXT.1.3 The application shall not download, modify, replace or update its own binary code.

FPT_TUD_EXT.1.4 Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.

FPT_TUD_EXT.1.5 The application is distributed [*with the platform OS*].

5.4.7 Class FTP: Trusted Path/Channel (FTP)

5.4.7.1 FTP_DIT_EXT.1 Protection of Data in Transit

FTP_DIT_EXT.1.1[11] The application shall encrypt all transmitted [*sensitive data*] using IPsec as specified in FCS_IPSEC_EXT.1 for [*VPN Tunnel*] and [

– *no other protocols*

] between itself and another trusted IT product.

5.5 Security Functional Requirements Drawn From the PP-Module

5.5.1 FCS: Cryptographic Support

5.5.1.1 FCS_CKM.1/VPN Cryptographic Key Generation (IKE)

FCS_CKM.1.1/VPN The TSF shall **[*implement functionality*] to** generate **asymmetric** cryptographic keys **used for IKE peer authentication** in accordance with: **[**

- *FIPS PUB 186-4, "Digital Signature Standard (DSS)," Appendix B.4 for ECDSA schemes and implementing "NIST curves," P-256, P-384 and [P-521]*

] and specified cryptographic key sizes [*equivalent to, or greater than, a symmetric key strength of 112 bits*] ~~that meet the following:~~ [**assignment:** *~~list of standards~~*].

5.5.1.2 FCS_CKM_EXT.2 Cryptographic Key Storage

FCS_CKM_EXT.2.1[12] The [*TOE platform*] shall store persistent secrets and private keys when not in use in platform-provided key storage.

5.5.1.3 FCS_CKM_EXT.4 Cryptographic Key Destruction

FCS_CKM_EXT.4.1[13] The [*TOE platform*] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

5.5.1.4 FCS_IPSEC_EXT.1 IPsec

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 implement [*tunnel mode*].

FCS_IPSEC_EXT.1.3 The TSF shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

FCS_IPSEC_EXT.1.4 The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms [*AES-GCM-128, AES-GCM-256 as specified in RFC 4106,* [*AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC*]].

FCS_IPSEC_EXT.1.5[14] The TSF shall implement the protocol: [

- *IKEv2 as defined in RFC 7296 (with mandatory support for NAT traversal as specified in section 2.23), RFC 8247, and [RFC 4868 for hash functions]*

[11] Modified as per MOD_VPNC_V2.4 and TD0753.
[12] As per TD0725
[13] As per TD0725
[14] As per TD0897.

].

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 as specified in RFC 6379 and* [*AES-GCM-128 as specified in RFC 5282, AES-GCM-256 as specified in RFC 5282*].

FCS_IPSEC_EXT.1.7 The TSF shall ensure that [

- *IKEv2 SA lifetimes can be configured by [an Administrator, a VPN Gateway] based on [length of time]*,

]. If length of time is used, it must include at least one option that is 24 hours or less for Phase 1 SAs and 8 hours or less for Phase 2 SAs.

FCS_IPSEC_EXT.1.8 The TSF shall ensure that IKE protocols implement DH Groups

- 19 (256-bit Random ECP), 20 (384-bit Random ECP) according to RFC 5114 and

[

- *[14 (2048-bit MODP), 15 (3072-bit MODP), 16 (4096-bit MODP)] according to RFC 3526,*
- *[21 (521-bit Random ECP)] according to RFC 5114*

].

FCS_IPSEC_EXT.1.9 The TSF shall generate the secret value x used in the IKE DH 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 [*256*, *384*, *512*] bits.

FCS_IPSEC_EXT.1.10 The TSF shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[*256*].

FCS_IPSEC_EXT.1.11 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 [*Pre-shared keys*].

FCS_IPSEC_EXT.1.12 The TSF shall not establish an SA if the [*IP address, Fully Qualified Domain Name (FQDN), user FQDN, Distinguished Name (DN)*] and [*no other reference identifier type*] contained in a certificate does not match the expected values for the entity attempting to establish a connection.

FCS_IPSEC_EXT.1.13 The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

FCS_IPSEC_EXT.1.14 The [*VPN Gateway*] 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.

5.5.2 Class FDP: User Data Protection

5.5.2.1 FDP_RIP.2 Full Residual Information Protection

FDP_RIP.2.1 The [*TOE platform*] shall ensure that any previous information content of a resource is made unavailable upon the [*allocation of the resource to*] all objects.

5.5.3 Class FIA: Identification and Authentication

5.5.3.1 FIA_PSK_EXT.1 Pre-Shared Key Composition

FIA_PSK_EXT.1.1 The TSF shall be able to use pre-shared keys for IPsec and [*no other protocols*].

FIA_PSK_EXT.1.2 The TSF shall be able to accept the following as pre-shared keys: [*generated bit-based*].

5.5.3.2 FIA_PSK_EXT.2 Generated Pre-Shared Keys

FIA_PSK_EXT.2.1 The TSF shall be able to [

- *accept externally generated pre-shared keys*

].

5.5.4 Class FMT: Security Management

5.5.4.1 FMT_SMF.1/VPN Specification of Management Functions (VPN)

FMT_SMF.1.1/VPN The TSF shall be capable of performing the following management functions: [

- *Specify client credentials to be used for connections*

]

5.5.5 Class FPT: Protection of the TSF

5.5.5.1 FPT_TST_EXT.1/VPN TSF Self-Test

FPT_TST_EXT.1.1/VPN The [*TOE platform*] shall run a suite of self tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.1.2/VPN The [*TOE platform*] shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the *[cryptographic services provided by the TOE platform's IMA]*.

5.6 Security Assurance Requirements

This section identifies the Security Assurance Requirements applicable to the TOE and the development environment. For conformance with the PP-Configuration, the Security Assurance Requirements are drawn from the Base-PP only. The applicable Security Assurance Components are stated in Table 17.

Table 17 Security Assurance Requirements

Security Assurance Class	Security Assurance Components
Security Target (ASE)	Conformance claims (ASE_CCL.1) Extended components definition (ASE_ECD.1) ST Introduction (ASE_INT.1) Security objectives for the operational environment (ASE_OBJ.1) Stated security requirements (ASE_REQ.1) TOE summary specification (ASE_TSS.1)
Development (ADV)	Basic functional specification (ADV_FSP.1)
Guidance Documents (AGD)	Operational user guidance (AGD_OPE.1) Preparative procedures (AGD_PRE.1)
Life Cycle Support (ALS)	Labelling of the TOE (ALC_CMC.1)

	TOE CM Coverage (ALC_CMS.1) Timely Security Updates (ALC_TSU_EXT.1)
Tests (ATE)	Independent testing - conformance (ATE_IND.1)
Vulnerability Assessment (AVA)	Vulnerability survey (AVA_VAN.1)

5.7 Security Requirements Rationale

The Security Functional Requirements are drawn from the Base-PP and PP-Module and not from any other source. The ST claims exact conformance to the Base-PP and to the PP-Module. The Security Functional Requirements include each mandatory requirement and each applicable optional and selection-based requirement. Only the operations allowed in the Base-PP and the PP-Module are implemented. Therefore, the Security Functional Rationales of the Base-PP and the PP-Module are directly applicable to the ST as well. They are not repeated here.

The Security Assurance Requirements are drawn from the Base-PP only as required by the PP-Configuration. None. are added or removed. Therefore, the Security Assurance Requirements Rationale of the Base-PP is directly applicable to the ST as well. It is not repeated here.

6 TOE Summary Specification

The TOE Summary Specification includes the description of how the TOE fulfills the security functional requirements, and how the developer and the evaluator fulfill the security assurance requirements. Each is described in this section. Additional details on the cryptographic algorithms and protocols implemented in the TOE are also given.

6.1 Fulfillment of the Security Functional Requirements Drawn from the Base-PP

The fulfilment of the Security Functional Components drawn from the Base-PP are given in Table 18. Each applicable Security Functional Component is identified, and the fulfilment of that component is described.

Table 18 Fulfilment of the Security Functional Components

<table>
<tr><th>Security Functional Component</th><th>Fulfilment</th></tr>
<tr><td>FCS_CKM.1
FCS_CKM.1/AK</td><td>The TOE implements key generation for asymmetric keys used by the Internet Protocol Security (IPsec). Three key generation methods are implemented:
– Elliptic Curve Cryptographic (ECC) schemes which use NIST curves P-256, P-384, and P-521. The ECC schemes for the generation of asymmetric keys is implemented in accordance with FIPS PUB 186-5, “Digital Signature Standard (DSS)”, Appendix B.4,
– FFC Schemes using safe-prime groups for the generation of ECC and RSA keys. FFC Schemes using safe-primes is implemented in accordance with NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”, RFC 3526, and RFC 7919.
This is implemented by the OpenSSL 1.1.1l library that’s included with the TOE.</td></tr>
<tr><td>FCS_CKM.2</td><td>The TOE implements key establishment for IPsec using the following key agreement methods:
– Elliptic curve-based key establishment schemes implemented in accordance with NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”, and
– Finite field-based key establishment schemes implemented in accordance with NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography".
This is implemented by the OpenSSL 1.1.1l library that’s included with the TOE.</td></tr>
<tr><td>FCS_COP.1/Hash</td><td>The TOE implements cryptographic hashing to support IKEv2 and IPsec. The Hash functions implemented are SHA-256, SHA-384, and SHA-512. Each is in conformance with FIPS Pub 180-4 “Secure Hash Standard.”
<table>
<tr><td></td><td>SHA-256</td><td>SHA-384</td><td>SHA-512</td></tr>
<tr><td>Block size</td><td>512 bits</td><td>1024 bits</td><td>1024 bits</td></tr>
<tr><td>Output size</td><td>256 bits</td><td>384 bits</td><td>512 bits</td></tr>
</table></td></tr>
<tr><td>FCS_COP.1/KeyedHash</td><td>The TOE implements HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512 to support message authentication on IKEv2 and IPsec. The details of the HMAC algorithms implemented are as follows:</td></tr>
</table>

<table>
<tr><td></td><td>
<table>
<tr><td></td><td>HMAC-SHA-256</td><td>HMAC-SHA-384</td><td>HMAC-SHA-512</td></tr>
<tr><td>Key length</td><td>256 bits</td><td>384 bits</td><td>512 bits</td></tr>
<tr><td>Hash function</td><td>SHA-256</td><td>SHA-384</td><td>SHA-512</td></tr>
<tr><td>Block size</td><td>512 bits</td><td>1024 bits</td><td>1024 bits</td></tr>
<tr><td>Output size</td><td>256 bits</td><td>384 bits</td><td>512 bits</td></tr>
</table>
</td></tr>
<tr><td>FCS_COP.1/Sig</td><td>The TOE implements cryptographic signature generation and verification for use with IPsec and IKEv2. Digital signature generation and verification is implemented using RSA and ECC Schemes:
– RSA schemes use cryptographic key sizes of at least 2048 bits. They are implemented in accordance with Sect. 5 of FIPS PUB 186-4, “Digital Signature Standard (DSS)”, and
– ECC-based schemes implement ECDSA using NIST curves P-256, P-384, and P-521. They are implemented in accordance with Sect. 6 of FIPS PUB 186-4, “Digital Signature Standard (DSS)”.
This is implemented by the OpenSSL 1.1.1l library that’s included with the TOE.</td></tr>
<tr><td>FCS_COP.1/SKC</td><td>The TOE implements symmetric encryption and decryption for IPsec and IKEv2 protocol payload using Advanced Encryption Standard (AES) in CBC and GCM modes:
– CBC mode is specified in NIST SP 800-38A, and
– GCM mode is specified in NIST SP 800-38D.
Key sizes of 128 bit and 256 bits are supported for both modes.</td></tr>
<tr><td>FCS_RBG_EXT.1
FCS_RBG_EXT.2</td><td>The application on the TOE shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using CTR_DRBG (AES). The implementation of the CTR DRBG is provided by the OpenSSL 1.1.1l library that’s included with the TOE.
The TOE will rely on the platform provided entropy source. This entropy source has been evaluated under NIST’s ESV Program and has been assigned the certificate #E142.
An additional Entropy Report that covers the Annex C requirements of PP_APP_V1.4 will be provided to the evaluators.</td></tr>
<tr><td>FCS_STO_EXT.1
FDP_DAR_EXT.1</td><td>The TOE does not implement secure non-volatile memory. Instead, the TOE stores all sensitive data in the non-volatile memories of the execution platform. The specific sensitive data of interest (and their purpose) are indicated in the table below. These are stored in an encrypted LUKS-encrypted container provided by the TOE platform. LUKS encrypts the container using AES-XTS. The TOE platform mounts this container at boot-time.
<table>
<tr><th>Sensitive Data</th><th>SFR</th><th>Purpose</th></tr>
<tr><td>X.509v3 certificates</td><td>FIA_X509_EXT.1
FIA_X509_EXT.2</td><td>Provides end-user identification for the TOE and the VPN Gateway.</td></tr>
<tr><td>Private key</td><td>FIA_X509_EXT.1
FIA_X509_EXT.2</td><td>Private key corresponding to the TOE’s certificate.</td></tr>
<tr><td>Connection templates</td><td>FMT_SMF.1/VPN</td><td>These contain the VPN-templates that define the</td></tr>
</table>
</td></tr>
</table>

			connections endpoints and the client certificates to use.

FDP_DEC_EXT.1 FDP_NET_EXT.1	The TOE implements only the VPN Client. It does not provide any general computing facilities and is not used for any other functions. The application only uses the network connectivity and the user PC or Laptop connectivity implemented by the platform: – Network connectivity is used to connect the TOE to the untrusted network. The TOE protects the communication over the untrusted network of the software executing in the user PC or Laptop. The primary means of protecting the communication is the VPN Connection between itself and the VPN Gateway of the Private Network. The user PC or laptop is only connected to the untrusted network through the TOE. – User PC or laptop connectivity is used for physically plugging the user PC or laptop to the TOE. The connection must be physical. No wireless connection to the TOE is supported. Not being a general-purpose computing platform and not implementing general services, the platform does not implement any sensitive information repositories the TOE could access. As such, the TOE does not access any sensitive information repositories.
FIA_X509_EXT.1	The TOE implements functionality and invokes functionality provided by the TOE platform to validate X.509 certificates used for IPsec connections. The X.509 certificates are validated using the certificate path validation algorithm defined in RFC 5280, which can be summarized as follows: – the public key algorithm and parameters are checked, – the current date/time is checked against the validity period, – the revocation status of the certificate is checked using CRL or OCSP, – the issuer name of X matches the subject name of X+1, and – any extensions are processed. The certificate validity check is performed when the TOE receives the certificate during an IPsec connection to the IPsec VPN Gateway: – The TOE invokes functionality provided by the TOE platform to ensure all CA certs contain the basic constraints extension and that the CA=TRUE flag is set. – The TOE invokes functionality provided by the TOE platform to ensure that the certificate path terminates in a trusted root CA (i.e. a CA certificate configured on the TOE as trusted). These checks ensure certificate validation results in a trusted root certificate. At any point if a certificate validation fails, the TOE shall not accept the certificate. The VPN Gateway is considered untrusted, and the TOE shall not establish the connection.
FIA_X509_EXT.2	When the TOE is provisioned, the organizational system administrator imports a certificate for the TOE in the non-volatile memory of the execution platform. When a certificate is validated, the TOE compares the FQDN of the server it is establishing connectivity with, against the Subject Alternate Name-dnsName attributes in the certificate. If there is a mismatch, the TOE shall not establish the connection. If a certificate cannot be successfully validated because of a network error, the TOE shall reject the certificate and not establish connection.

<table>
<tr><td>FMT_MEC_EXT.1</td><td>The TOE attempts to establish the VPN connection in accordance with the IPsec templates. The templates are stored in the non-volatile memory of the execution platform at the provisioning of the TOE by the organizational system administrator. The guidance of the templates may be overruled by a successfully identified and authenticated VPN Gateway.

Further, when the execution platform is deployed, the organizational system administrator must set the following parameters:

– "FIPS Mode",
– "Strict Certificate Trust", and
– "Enable CRL Check".</td></tr>
<tr><td>FMT_CFG_EXT.1</td><td>The TOE uses three types of sensitive information stored at the non-volatile memory of the execution platform at the provisioning of the TOE:

– X.509 certificate of the VPN Client,
– Pre-shared cryptographic keys used for IPsec, and
– IPsec templates which define the IPsec parameters the TOE attempts to negotiate with the VPN Gateway.

Absence of any of the above shall result in the denial of the service for which the credentials are required. The TOE shall only be granted access to the TOE functions through the above means. All credentials are stored for the use by the TOE at the provisioning and cannot be modified or generated by the user. Users can only access files which are associated to the installation that user performed and cannot modify the provisioning files.</td></tr>
<tr><td>FMT_SMF.1</td><td>The TOE does not implement any management functions. All configuration of the TOE is done at the provisioning by the organisational system administrator through the management interface of the execution platform. Once the deployment is completed, there are no user-accessible functions for managing the TOE.</td></tr>
<tr><td>FPR_ANO_EXT.1</td><td>The TOE does not store or transmit any Personally Identifiable Information.</td></tr>
<tr><td>FPT_API_EXT.1</td><td>The platform API used by the TOE are listed in the categories below.

Cryptographic APIs

<table>
<tr><th>API</th><th>Description</th></tr>
<tr><td>getrandom()</td><td>Retrieve random bytes and fill the buffer provided by user.</td></tr>
<tr><td>crypto_alloc_aead() / crypto_free_aead()</td><td>AEAD cipher allocation</td></tr>
<tr><td>crypto_alloc_ahash() / crypto_free_ahash()</td><td>Hash algorithm allocation</td></tr>
<tr><td>crypto_alloc_skcipher() / crypto_free_skcipher()</td><td>Symmetric cipher allocation</td></tr>
<tr><td>crypto_aead_encrypt() / crypto_aead_decrypt()</td><td>AEAD operations</td></tr>
<tr><td>crypto_ahash_digest()</td><td>Hash digest computation</td></tr>
<tr><td>crypto_skcipher_encrypt() / crypto_skcipher_decrypt()</td><td>Symmetric encryption</td></tr>
</table>
</td></tr>
</table>

Dynamic Loading APIs

API	Description
dlopen()	Load shared library
dlsym()	Get symbol address
dlclose()	Unload shared libraries
dlerror()	Get dynamic loading errors
dladdr()	Get symbol information

Environment

API	Description
getenv()	Retrieve environment variables from platform

Error Handling

API	Description
strerror()	Converts error codes to strings
perror()	Print error messages.

File System Operations

API	Description
open()	Opening files and devices
read()	Reading from files and devices
write()	Writing to files
close()	Closing file descriptors
fopen()	Standard C library file operations
stat() / lstat() / fstat()	File status operations
access()	File accessibility checks
mkdir()	Create directories
mmap()	Memory mapping files (for efficient file access)
munmap()	Unmapping memory mapped files
opendir()/readdir()/closedir()	Directory operations.

Sysctl and Proc APIs

API	Description

register_net_sysctl()	Network namespace sysctl registration
proc_create() / proc_remove()	/proc filesystem operations
seq_printf() / seq_putc()	Sequential file operations

Memory Management

API	Description
__get_free_pages()/free_pages()	Page allocation (and deallocation) within the kernel space.
alloc_skb()/kfree_skb()	Socket buffer allocation (and deallocation)
skb_clone()/skb_copy()	Socket buffer cloning and copying.
malloc() / calloc() / realloc()	Memory allocation
pskb_expand_head()	Socket buffer header expansion.
free()	Memory deallocation
kmalloc()/kfree()	Dymanic memory allocation within the kernel space
kmemdup()	Memory duplication within the kernel space.
kzalloc()	Zero-initialized memory allocation in the kernel space.
malloc_usable_size()	Retrieve the usable memory size
mmamp()	Create mapping in the virtual address space.
mlock() / munlock()	Locking memory pages (for secure memory)
munmap()	Unmap mapped memory.
mprotect()	Memory protection
brk() / sbrk()	Process memory management

POSIX Threading API and Kernel Threading

API	Description
pthread_join()	Thread joining
pthread_detach()	Thread detachment
pthread_cancel()	Thread cancellation
pthread_kill()	Send signals to threads
pthread_rwlock_init/destroy/rdlock/wrlock/unlock()	Read
pthread_mutex_init/destroy/lock/unlock()	Mutexes
pthread_cond_init/destroy/wait/signal/broadcast()	Condition variables
pthread_once()	One

pthread_cleanup_push/pop()	Cleanup handlers
pthread_setcancelstate/setcanceltype()	Thread cancellation control
pthread_setname_np()	Set thread names.
spin_lock_bh() / spin_unlock_bh()	Bottom-half spinlocks
rcu_read_lock() / rcu_read_unlock()	RCU read-side critical sections
mutex_lock() / mutex_unlock()	Mutex operations
refcount_inc() / refcount_dec_and_test()	Reference counting

Process and System Information

API	Description
getenv()	Retrieve environment variables
getpid()	Retrieve process ID
gettid()	Get thread ID
getuid()/geteuid()	Retrieve user IDs
getguid()/geteuid()	Get group IDs
getpagesize()	Retrieve number of bytes of memory in a page.
setenv()/putenv()	Set environment variables
setlocale()	Set or retrieve the current locale.
setuid()/seteuid()	Set user IDs
setuid()/seteuid()	Set user IDs
setgid()/setegid()	Set group IDs
sched_yield()	Yield CPU to other threads
sysconf()	Retrieve operating system page-size
clock_gettime()	Current time in max resolution as supported by platform.
gettimeofday()	Current time.
uname()	Retrieve system information.

Signal Handling APIs

API	Description
singal()	Set-up signal handler
sigaction()	Change signal action on receipt of signal
kill()	Send signal to any process group or process.
sigprocmask()	Fetch or change the signal mask of the calling process

sigsuspend()	Suspend a process until a signal is received.

Socket, Network and Packet Processing APIs

API	Description
netdev_features_t	Network device feature handling
register_netdev() / unregister_netdev()	Network device registration
netlink_has_listeners()	Netlink socket checking
netlink_kernel_create()	Netlink socket creation
skb_dst() / skb_dst_drop() / skb_dst_set()	Destination handling
dst_clone() / dst_release()	Destination reference counting
ip_hdr() / ipv6_hdr()	IP header access
skb_network_header() / skb_transport_header()	Header positioning
socket()	Create sockets (AF_INET, AF_INET6, AF_UNIX)
bind()	Bind sockets to addresses
connect()	Connect to remote endpoints
listen()	Listen for connections
accept()	Accept incoming connections
send()/sendto()/sendmsg()	Send data
recv()/recvfrom()/recvmsg()	Receive data
close()	Close socket descriptors
poll()	Poll for socket events
setsockopt()/getsockopt()	Set and get socket options
nlmsg_new() / nlmsg_free()	Netlink message allocation
nla_put() / nla_get()	Netlink attribute handling
genl_register_family()	Generic netlink family registration
genlmsg_put() / genlmsg_end()	Generic netlink message construction
get_net() / put_net()	Network namespace reference counting
net_generic()	Per-network namespace data access
register_pernet_subsys()	Per-network namespace registration
nf_hook()	Netfilter hook invocation
__skb_pull() / skb_push()	Socket buffer data manipulation
skb_headroom() / skb_tailroom()	Buffer space checking

skb_set_inner_transport_header()	Inner header setting

Flow and Routing APIs

API	Description
flowi	structure handling flow identification
fib_lookup()	Forwarding information base lookup
ip_route_output_flow()	IPv4 route output
xfrm_dst_lookup()	XFRM destination lookup

Security Policy Database APIs

API	Description
xfrm_policy_alloc() / xfrm_policy_destroy()	Policy management
xfrm_state_alloc() / xfrm_state_delete()	State management

Time and Clock APIs

API	Description
gettimeofday()	Retrieve current time
clock_gettime()	Retrieve current time of a specified clock
nanosleep()	High resolution sleep
time()	Get time in seconds
localtime()/gmtime()	Time conversion
timer_setup() / mod_timer() / del_timer()	Timer management
schedule_work()	Work queue scheduling
tasklet_schedule()	Tasklet scheduling
jiffies	Time keeping

Hardware Offload APIs

API	Description
netdev_features_t	handling for NETIF_F_HW_ESP
xfrmdev_ops	structure callbacks
validate_xmit_xfrm()	Transmit validation for offload

Module and Initialisation APIs

API	Description
module_init() / module_exit()	Module initialization

	subsys_initcall() | Subsystem initialization
FPT_AEX_EXT.1	The compiler flags used to enable ASLR when the TOE is compiled are: `-fPIE -pie` for executables and `-fPIC -shared` for shared libraries. The compiler flag used to enable stack-based buffer overflow protection in the TOE is: `-fstack-protector-strong` The TOE does not allocate any memory region with both write and execute permissions
FPT_IDV_EXT.1	The TOE implements a version number of form X.Y where X is the major release number and Y is the minor release number. For each minor release, the minor release number is incremented by one. For each major release, the major release number is incremented by one.
FPT_LIB_EXT.1	The TOE requires the following libraries. The following libraries are provided by the TOE platform: - libcurl4-8.0.1 - libgcrypt20-1.9.4 (functionality provided by the library is explicitly disabled) - libxml2-2-2.9.14 - glibc-2.31 - libsystemd0-249.17 Provided with the TOE: - openssl-1.1.1l - fipscheck-1.4.1
FPT_TUD_EXT.1	The application is distributed with the underlying Ad Noctem GB-100 Operating System image (i.e. the TOE Platform). The application can only be updated through updates to the underlying TOE Platform. The update image to the underlying TOE platform is signed using GPG with an RSA 4096 public key and SHA-256 hash by Northrop Grumman Corporation. Northrop Grumman Corporation are the authorised source for any updates to the TOE. This is verified by the update script before the TOE Platform is updated. Current version of the TOE can be verified through the "System Overview" screen.
FTP_DIT_EXT.1	The TOE implements IPsec between itself and the VPN Gateway. That connection is used to protect any communication between the applications executing in the user PC or Laptop and the resources of the Private Network.

6.2 Fulfillment of the Security Functional Requirements Drawn from the PP-Module

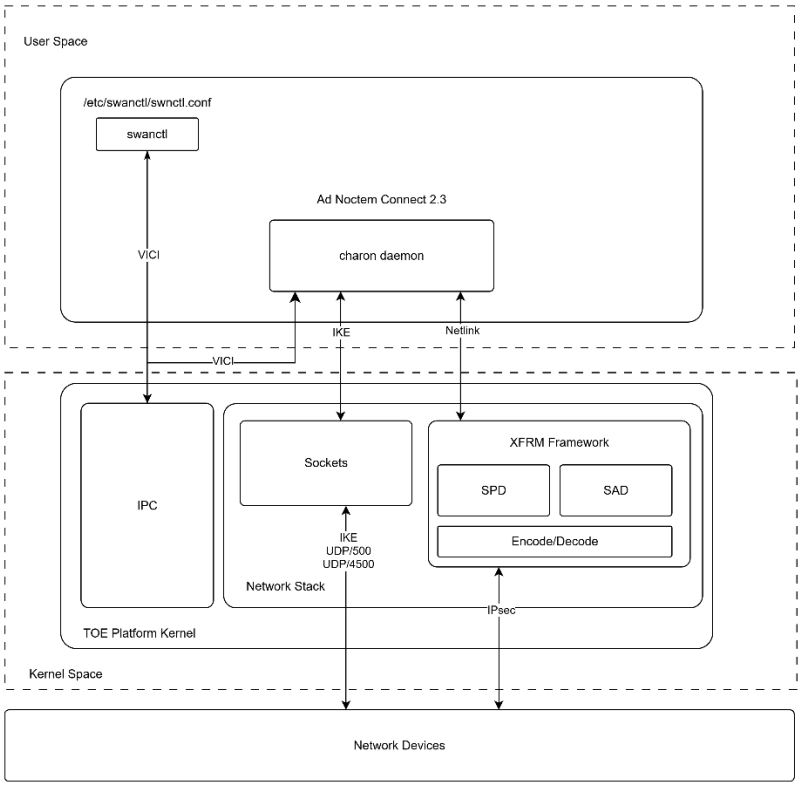
The fulfilment of the Security Functional Components drawn from the PP-Module is given in Table 19. Each applicable Security Functional Component is identified, and the fulfilment of that component are described.

Table 19 Fulfillment of the Security Functional Components Drawn from the PP-Module

FCS_CKM.1/VPN	The TOE generates the VPN keys for the use in the IKE handshake. The TOE Platform provides a specified key generation algorithm to generate asymmetric cryptographic keys for Internet Key Exchange (IKE) authentication. The key sizes are NIST curve sizes P-256, P-384 or P-521 when Elliptic-curve

	Diffie–Hellman (ECDH) is used. The key generation function is invoked by the TOE when the TOE is configured to use ECDH for IKE.
FCS_CKM_EXT.2	The TOE does not implement any secure memory. It uses the platform to store the RSA and ECDSA private keys used for IKE peer authentication in the non-volatile memory. The TOE platform implements disk encryption. The TOE does support RFC 8784 Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security.
FCS_CKM_EXT.4	Ephemeral cryptographic key material is held in RAM and is cleared by removing power from the RAM. For non-volatile memory, information like the x.509 certificates are indefinite. An administrator of the platform may manually destroy them. To erase long-term key material held in files the platform provides the tool `fstrim`. After a deletion of a file with sensitive data, this tool uses the SSD TRIM command to inform the SSD to discard unused blocks bypassing wear leveling. The tool `shred` is available that overwrites files multiple times with random data. This tool can be used by the administrator of the platform to delete data from the HDD.
FCS_IPSEC_EXT.1	In the evaluated configuration, the TOE is only configured to support IKEv2. IKE separates negotiation into two phases: Phase 1 and Phase 2. During IKE Phase 1, the TOE is able to authenticate the remote VPN Gateway using device-level authentication with RSA or ECDSA X.509v3 certificates provided by the TOE platform. The TOE is also able to use pre-shared keys for this purpose, if configured appropriately. The TOE compares its reference identifier to the identifier presented by the VPN Gateway peer. The TOE supports reference identifiers as configured by the Administrator to be either FQDN or IP address and compares it to the Subject Alternative Name (SAN) or the Common Name (CN) fields in the certificate of the peer. The order of comparison is SAN followed by CN. If the TOE successfully matches the reference identifier to the presented identifier, IKE Phase 1 authentication will succeed. Otherwise, it will fail if it does not match. Phase 1 creates the first tunnel, which protects later IKE negotiation messages. The key negotiated in phase 1 enables IKE to communicate securely in phase 2. The TOE supports only IKEv2 session establishment. The TOE supports Diffie-Hellman Group 19 (256-bit Random ECP), 20 (384-bit Random ECP) and 21 (521-bit Random ECP) in support of IKE Key Establishment negotiated in Phase 1. These keys are generated using the DRBG specified in FCS_RBG_EXT.1 having 256 bits of entropy. It is the responsibility of the VPN Gateway to ensure by default that the strength of the symmetric algorithm used for IKEv2_SA is greater than the strength of the symmetric algorithm used for IKEv2 CHILD_SA. Once each phase is established, the communication through the tunnels is encrypted. All cryptographic operations that occur at the IKE-level are handled by the TOE. The TOE uses OpenSSL 1.1.1l to carry out these operations.

	In the evaluated mode of operation, the TOE supports: AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256 ciphers. For authentication, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512 are supported. The TOE platform implements the IPsec standard according to RFC 4301. It uses the Encapsulating Security Payload (ESP) protocol (RFC 4303) to provide confidentiality, data origin authentication, connectionless integrity and anti-replay protection. The ESP protocol operates in tunnel mode by default, and no configuration is required for this to be enabled. The IPsec VPN Gateway administrator is responsible for managing remote access policies that offer an interface for creating Access Control Lists (ACLs) to specify which network segments need IPsec protection. By default, the remote access policy configures the TOE to secure all network traffic using IPsec. The Security Policy Database (SPD) and the Security Association Database (SAD), as specified in RFC 4301, is implemented within the XFRM framework that's part of the TOE platform. The SPD consists of policy rules (SPD entries) that are represented by a C structure, struct xfrm_policy, which define security policies for network traffic. The main APIs used for SPD lookup is xfrm_lookup(), which look for matches based on destination and source IP addresses, source and destination port addresses, protocol, and interface index. If no policy match is found, the TOE will drop the packet. When a policy match is found, the TOE platform performs an XFRM state lookup using information from the matched policy to identify the correct Security Association (SA) from the SAD. The SAD is represented internally by a C structure, struct xfrm_state. The information SA is used encrypt, decrypt or transform a received packet. The TOE communicates with the XFRM framework using Netlink sockets (specifically NETLINK_XFRM) to create, modify, and delete SPD entries. This is managed by the kernel-netlink plugin. The plugin: translates TOE configuration to XFRM policies, manages Security Association (SA) installation and deletion and handles policy priorities and selectors. The TOE can set a policy to apply to matching source and destination addresses through the swanctl.conf file. The PROTECT, BYPASS and DISCARD rules are configured in this way. The `mode` configuration key in swanctl.conf (or the inherited configuration) can be set to `pass` for BYPASS, `drop` for DISCARD and `tunnel` for protect. Priority for the processing of these rules are according to the underlying XFRM policy priorities: • More specific policies get higher priorities • PROTECT policies have higher priorities than BYPASS. • DISCARD policies have the lowest priority. XFRM processes policies in priority order with first match determining the action.

	A diagram showing the interaction between the TOE and the TOE platform in implementing the IKE and IPsec functionality is shown below.  The encode and decode operations within the XFRM framework apply the cryptographic operations to a packet traversing the framework as per requirement of the ESP protocol. The TOE platform implements the cryptographic operations that will be applied. The TOE platform uses the Linux Kernel Crypto API to implement these cryptographic operations. In the evaluated configuration, the TOE supports AES-CBC-128 and AES-CBC-256 and AES-GCM-128 and AES- GCM-256 modes of encryption. For authentication, when the CBC mode of encryption is used, the TOE allows for HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512.
FDP_RIP.2	When the TOE generates protocol data units for IPsec or IKE, all padding is with zeros. The TOE explicitly uses zeros to pad the protocol data units. This ensures that whenever a new data unit is generated, no information of the previous data units is used.
FIA_PSK_EXT.1 FIA_PSK_EXT.2	The TOE allows use of pre-shared keys in the establishment of the VPN tunnel. The pre-shared keys are generated in the deployment environment of the TOE by the organizational administrator. They are stored in a bit-based representation in the secure memory of the platform. If the IPsec template selected by the user uses a pre-shared key, the TOE shall use the PSK to negotiate a IKEv2 tunnel.
FMT_SMF.1/VPN	The TOE does not implement any management interface when operational. All management functions are performed by the organizational system administrator when the TOE is being deployed.

	The two VPN management functions may be carried out by the organizational system administrator: – The administrator may specify the TOE as an IPsec-capable network device to use for VPN connections. This requires configuring the platform so that the X.509v3 certificates, pre-shared keys, and IPsec templates are consistent with the VPN Gateway and the VPN policies of the organization operating the private network. – The administrator may specify client credentials to be used for connections. The credentials are the X.509v3 certificates and the pre-shared keys stored in the memories of the platform.
FPT_TST_EXT.1/VPN	During platform bootup, the TOE runs cryptographic self-tests on: - The kernel cryptographic library - OpenSSL 1.1.1 The TOE is shipped with pre-generated HMAC-SHA256 checksum files for the TOE, the TOE Platform Kernel and OpenSSL 1.1.1l. These are verified using the fipscheck utility that's provided by the TOE Platform during boot-up. The fipscheck utility uses the cryptographic implementations provided by OpenSSL 1.1.1l library to verify the pre-computed checksums.
ALC_TSU_EXT.1	The customer is directly contacted by NGC when security updates are available via email. It is the responsibility of the organizational system administrator to download and apply the update to the TOE and TOE Platform. If any configuration updates are present that affect the TOE, the organizational system administrator will be provided with additional documentation detailing the change. Security vulnerabilities can be reported to NGC via the contact information provide in the TOE Security Guidance.

6.3 Fulfillment of the Security Assurance Requirements

To fulfill the Security Assurance Requirements, the developer implements a set of security assurance measures. Some assurance classes are fulfilled by the evaluator of the TOE. The security assurance measures implemented by the developer and evaluator of the TOE are described in Table 20.

Table 20 Fulfillment of the Security Assurance Requirements

Security Assurance Requirement	Fulfilment
Security Target	The developer authors a Common Criteria Security target for the Target of Evaluation. The Security Target implements all assurance components required by the Base-PP. The Security Target includes – A ST Introduction which provides a ST Reference, a TOE Reference, a TOE Overview, and a TOE Description. – Conformance Claims stating exactly the conformance to the Common Criteria and the Protection Profiles, Protection Profile Modules and Protection Profile Configurations the Security Target and the Target of Evaluation claim conformance to.

	– A Security Problem Definition which is a statement of Threats, Assumptions and Organizational Security Policies applicable to the TOE. – A statement of the security objectives for the TOE. The Base-PP only defines security requirements for the operational environment of the TOE, but the Security Target also states the security requirements for the TOE drawn from the PP-Module. – Extended Components Definition and the statement of the security requirements state exactly the Security Functional Requirements and the Security Assurance Requirements the TOE fulfills. – TOE Summary Specification which describes for each Security Functional Requirement how the TOE fulfills that Security Functional Requirement.
Functional Specification	Included in the TOE Summary Specification, the developer provides all information required for a basic functional specification of the TOE.
Security Guidance	Attached to the TOE and included in the physical scope of the TOE is a Common Criteria Guidance Supplement for the TOE. The Guidance Supplement gives guidance to the user of the TOE in the secure installation and preparation of the TOE so that the TOE is in an initial secure state. The Guidance Supplement also provides guidance to the user of the TOE so that the TOE always remains in a secure state when the guidance is followed.
Life Cycle Support	The developer labels the TOE with a unique identifier. The label may be examined by the user of the TOE to ensure that the correct version of the TOE is used. When the TOE software is updated, the label of the TOE is updated accordingly. TOE upgrades are done in accordance with a well-defined policy governing the updating. The TOE label is included in the configuration list of the TOE to ensure that the evaluator can be assured of evaluating the intended version of the TOE.
Independent Testing	The evaluator carries out a set of independent tests on the TOE. The independent tests complement the functional testing carried out by the developer and ensure that the TOE passes each applicable test required for conformance with the Base-PP, PP-Module and Functional Package. The evaluator documents the testing in accordance with the requirements stated in the Base-PP, the PP-Module, the Functional Package and the Common Criteria evaluation and certification scheme followed.
Vulnerability Assessment	The evaluator carries out a vulnerability survey to determine that there are no obvious vulnerabilities in the TOE which could be practically exploited by the threat agents. The evaluator documents the vulnerability survey in accordance with the requirements stated in the Base-PP, the PP-Module, the Functional Package and the Common Criteria evaluation and certification scheme followed.

6.4 Cryptographic Details and CAVP References

This section provides additional details on the cryptographic algorithms and protocols implemented by the TOE.

Table 21 – Ad Noctem Connect Library

Certificate	Algorithms
A7623	AES-CBC AES-CCM AES-CTR AES-ECB AES-GCM SHA-1 SHA2-256 SHA2-384 SHA2-512 CTR-DRBG ECDSA KeyGen ECDSA KeyVer ECDSA SigGen ECDSA SigVer HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 RSA SigGen RSA SigVer KAS-ECC-SSC Sp800-56Ar3 KAS-FFC-SSC Sp800-56Ar3 KDF IKEv2 Safe Primes Key Generation Safe Primes Key Verification

Table 22 – Ad Noctem Kernel Cryptographic Library

Certificate	Algorithms
A7624	AES-CBC AES-CCM AES-GCM HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512 SHA-1 SHA2-256 SHA2-384 SHA2-512

7 Acronyms

AES	Advanced Encryption Standard
CA	Certificate Authority
CAVP	Cryptographic Algorithm Validation Program
CBC	Cipher Block Chaining
CC	Common Criteria
CCEVS	Common Criteria Evaluation and Validation Scheme
CEM	Common Evaluation Methodology
CLI	Command Line Interface
CMVP	Cryptographic Module Validation Program
CRL	Certificate Revocation List
CSfC	Commercial Solutions for Classified
CSP	Critical Security Parameter
CSR	Certificate Signing Request
CTR	Counter Mode
DHCP	Dynamic Host Configuration Protocol
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
ECDSA	Elliptic Curve Digital Signature Algorithm
EMI	Electromagnetic Interference
ESP	Encapsulating Security Payload
FFC	Finite Field Cryptography
FIPS	Federal Information Processing Standard
FIPS PUB	FIPS Publication
GCM	Galois Counter Mode
HMAC	Hash-Based Message Authentication Code
ICMP	Internet Control Message Protocol
IKE	Internet Key Exchange
IKE_SA	Internet Key Exchange Security Association
IKEv1	Internet Key Exchange version 1
IKEv2	Internet Key Exchange version 2

IP	Internet Protocol
IPS	Intrusion Prevention System
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISO	International Organization for Standardization
IV	Initialization Vector
KDF	Key Derivation Function
KW	Key Wrap
LAN	Local Area Network
MAC	Message Authentication Code
MDG	Management Daemon
NAT	Network Address Translation
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OCSP	Online Certificate Status Protocol
OID	Object Identity
OS	Operating System
OSP	Organizational Security Policy
PAM	Pluggable Authentication Modules
PFE	Packet Filtering Engine
PKCS	Public Key Cryptography Standard
PKI	Public Key Infrastructure
PRF	Pseudorandom Function
RFC	Request For Comments
RBG	Random Bit Generator
SA	Security Association
SD	Supporting Document
SHA	Secure Hash Algorithm
SSH	Secure Shell
SSHD	SSH Daemon
TLS	Transport Layer Security

TSF	TOE Security Function
TTL	Time To Live
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VPN	Virtual Private Network
XPN	Extended Packet Numbering