Australian Government
Department of Defence

ASD-Approved Protection Profile

Protection Profile for IPsec Virtual Private Network (VPN) Clients

Version:	**1.4**
Technology type:	**Network Related Devices**
Authored by:	**NIAP, United States**
Publication date:	**October 2013**
ASD approval date:	**30 May 2016**

The following document is a Protection Profile authored by NIAP, United States. This Protection Profile has been approved for use by the Australian Signals Directorate.

This Protection Profile (PP) describes the security requirements IPsec Virtual Private Networks (VPN) Clients. Compliant TOEs will provide security functionality that addresses threats to the TOE and implements policies that are imposed by law or regulation.

This PP in describing security requirements for IPsec Virtual Private Networks (VPN) Clients is intended to provide a minimal, baseline set of requirements that are targeted at mitigating well defined and described threats.

This PP specifies Security Functional Requirements (SFRs) for a VPN Client. A VPN provides a protected transmission of private data between VPN Clients and VPN Gateways. The TOE defined by this PP is the VPN Client, a component executing on a remote access client, using a platform API that enables the VPN client application to interact with other applications and the client device platform (part of the Operational Environment of the TOE). The VPN Client is intended to be located outside or inside of a private network, and provides a secure tunnel to a VPN Gateway. The tunnel provides confidentiality, integrity, and data authentication for information that travels across the public network. All VPN clients that comply with this document will support IPsec.

For information relating to the application of Protection Profiles, please refer to the Australian Government Information Security Manual (ISM) or www.asd.gov.au Controls in the ISM take precedence over any requirements contained in this PP where there is a conflict.

Protection Profile for IPsec Virtual Private Network (VPN) Clients

21 October 2013

Version 1.4

Table of Contents

List of Tables

List of Figures

Revision History

Version	Date	Description
1.0	December 2011	Initial release
1.1	December 2012	Minor update. To make cryptographic requirements consistent with the VPN Gateway Extended Package.
1.2	January 2013	Updated **FCS_COP.1.1(2)** to make consistent with the VPN Gateway Extended Package.
1.3	April 2013	Updated X.509 requirements to specify the certificate path validation algorithm must ensure a basicConstraints field is present and the cA flag set to TRUE as a condition that must be met for a certificate to be considered a CA certificate.
1.4	October 2013	Modifications to bring PP requirements into line with those in the Mobile Device PP. Structure selected requirements and assurance activities to reflect whether they need to be satisfied by the TOE, the operational environment (the platform), or either.

1 Introduction to the PP

This Protection Profile (PP) supports procurements of commercial off-the-shelf (COTS) IPsec Virtual Private Network (VPN) Clients to provide secure tunnels to authenticated remote endpoints or gateways. This PP details the policies, assumptions, threats, security objectives, security functional requirements, and security assurance requirements for the VPN and its supporting environment.

The primary intent is to clearly communicate to developers the Security Functional Requirements needed to counter the threats that are being addressed by the VPN Client. The description in the TOE Summary Specification (TSS) of the Security Target (ST) is expected to document the architecture of the product (Target of Evaluation) and the mechanisms used to ensure that critical security transactions are correctly implemented.

1.1 Overview of the TOE

This document specifies Security Functional Requirements (SFRs) for a VPN Client. A VPN provides a protected transmission of private data between VPN Clients and VPN Gateways. The TOE defined by this PP is the VPN Client, a component executing on a remote access client, using a platform API that enables the VPN client application to interact with other applications and the client device platform (part of the Operational Environment of the TOE). The VPN Client is intended to be located outside or inside of a private network, and provides a secure tunnel to a VPN Gateway. The tunnel provides confidentiality, integrity, and data authentication for information that travels across the public network. All VPN clients that comply with this document will support IPsec.

1.1.1 Usage and major security features of TOE

A VPN Client allows remote users to use client computers to establish an encrypted IPsec tunnel across an unprotected public network to a private network (see Figure 1). The TOE sits between the public network and entities (software, users, etc.) that reside on the VPN Client's underlying platform. IP packets crossing from the private network to the public network will be encrypted if their destination is a remote access VPN Client supporting the same VPN policy as the source network. The VPN Client protects the data between itself and a VPN Gateway, providing confidentiality, integrity, and protection of data in transit, even though it traverses a public network.

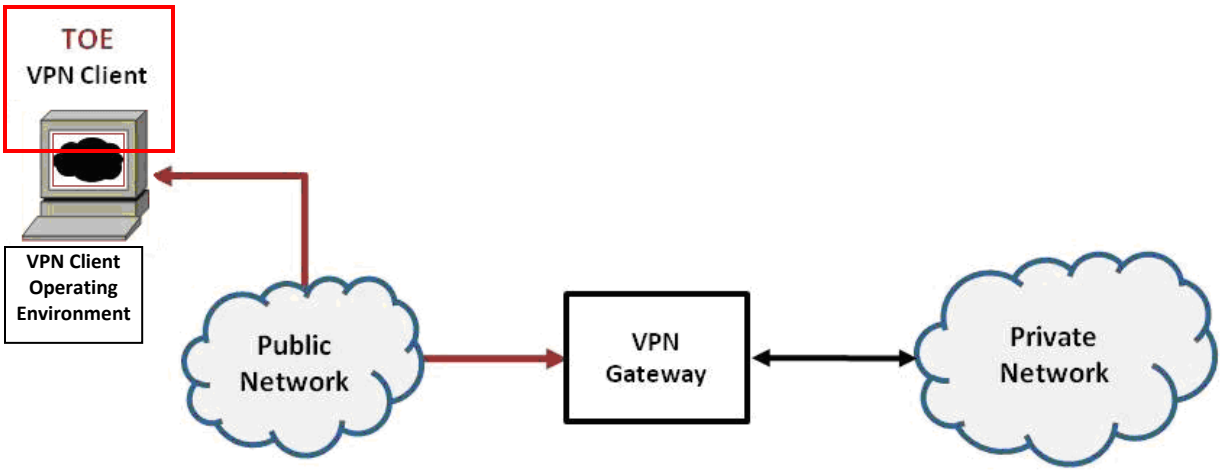


Figure 1: VPN Client

The focus of the Security Functional Requirements in this PP is on the following fundamental aspects of a VPN Client:

- Authentication of the VPN Gateway;
- Cryptographic protection of data in transit; and
- Implementation of services.

A VPN client can establish VPN connectivity with another VPN endpoint client or a VPN Gateway (that is the "remote" endpoint in the VPN communication). VPN endpoints authenticate each other to ensure they are communicating with an authorized external IT entity. Authentication of a VPN Gateway is performed as part of the Internet Key Exchange (IKE) negotiation. The IKE negotiation uses a pre-existing public key infrastructure for authentication and can optionally use a pre-shared key. When IKE completes, an IPsec tunnel secured with Encapsulating Security Payload (ESP) is established.

It is assumed that the VPN Client is implemented properly and contains no critical design mistakes. The VPN Client relies on the operational environment for its proper execution. The vendor is required to provide configuration guidance (AGD_PRE, AGD_OPE) to correctly install and administer the client machine and the TOE for every operational environment supported.

1.1.2 Cryptography

The IPsec VPN Client enables encryption of all information that flows between itself and its VPN Gateway. The VPN Client serves as an endpoint for an IPsec VPN tunnel and performs a number of cryptographic functions related to establishing and maintaining the tunnel. If the cryptography used to authenticate, generate keys, and encrypt information is sufficiently robust and the implementation has no critical design mistakes, an adversary will be unable to exhaust the encryption key space to obtain the data. Compliance with IPsec standards, use of a properly seeded Random Bit Generator (RBG), and secure authentication factors will ensure that access to the transmitted information cannot be obtained with less work than a full exhaust of the key space. Any plaintext secret and private keys or other cryptographic security parameters will be zeroized when no longer in use to prevent disclosure of security critical data.

1.1.3 Operational Environment

The TOE supporting environment is significant. The TOE is purely a software solution executing on a ”platform” (some sort of operating system running on a set of hardware). As such, the TOE must rely heavily on the TOE Operational Environment (system hardware, firmware, and operating system) for its execution domain and its proper usage. The vendor is expected to provide sufficient installation and configuration instructions to identify an Operational Environment with the necessary features and to provide instructions for how to configure it correctly.

The PP contains requirements (Section 4) that must be met by either the TOE or the platform on which it operates. A “platform” is defined as a separate entity whose functions may be used by the TOE, but is not part of the TOE. A third-party library compiled-in to the TOE is not considered part of the TOE’s “platform”, but (for instance) cryptographic functionality that is built in to an Operating System on which the VPN client executes can be considered part of the platform.

This is somewhat different than other PPs, but addresses most implementations of VPN clients where some part of the functionality of the IPsec tunnel is provided by the platform. In terms of the cryptographic primitives (random number generation, encryption/decryption, key generation, etc.) it is actually desirable that a well-tested implementation in the platform is used rather than trying to implement these functions in each client.

TOEs conformant with this PP must list the specific applicable platforms in the ST. The platforms will need to be identified to the point that they can be compared with a list of previously evaluated platforms (e.g., General Purpose Operating Systems, Network Devices, Mobile Devices) so that the implementation of the requirements not done by the TOE itself can be verified.

Requirements that can be satisfied by either the TOE or the platform are identified in Section 4 by text such as “The [selection: TSF, platform] shall…”. The ST author will make the appropriate selection based on where that element is implemented. It is allowable for some elements in a component to be implemented by the TOE, while other elements in that same component be implemented by the platform (requirements on the usage of X509 certificates is an example of where this might be the case, where using the information contained in the certificates and the implementation of revocation checking may be done by the TOE, but storage and protection of the certificates may be done by the platform).

For these requirements, there are two sets of assurance activities. Assurance activities will differ based on where the function that meets the requirement is implemented. In most cases, requirements implemented by the platform will require that the evaluator examine documents pertaining to the platform (generally the ST), while requirements implemented by the TOE may require examination of the TSS, examination of the Operational Guidance, or testing to be performed by the evaluator. For requirements implemented by the platform there may also be requirements that the evaluators examine the interfaces used by the TOE to access these functions on the platform to ensure that the functionality being invoked to satisfy the requirements of this PP is the same functionality that was evaluated.

1.1.4 Protocol Compliance

The TOEs meeting this PP will implement the Internet Engineering Task Force (IETF) Internet Protocol Security (IPsec) Security Architecture for the Internet Protocol, RFC 4301, as well as the IPsec

Encapsulating Security Payload (ESP) protocol. IPsec ESP is specified in RFC 2406 and RFC 4303. The IPsec VPN Client will support ESP in either tunnel mode, transport mode, or both modes.

The IPsec VPN Client will use either the Internet Key Exchange (IKE)v1 protocol as defined in RFCs 2407, 2408, 2409, 4109 or the IKEv2 protocol as specified in RFCs 5996 (with mandatory support for NAT traversal as specified in section 2.23), and 4307 to authenticate and establish session keys with the VPN entities.

In order to show that the TSF implements the RFCs correctly, the evaluator shall perform the assurance activities documented in this PP. In future versions of this PP, assurance activities may be augmented, or new ones introduced that cover more aspects of RFC compliance than is currently described in this publication.

2 Security Problem Definition

This PP is written to address the situation in which a remote user uses a public network to access a private network (e.g., the user's office network). Protection of network packets is desired as they cross the boundary between a public network and a private network. To protect the data in-transit from disclosure and modification, a VPN is created to establish secure communications. The VPN Client provides one end of the secure VPN tunnel and performs encryption and decryption of network packets in accordance with a VPN security policy negotiated between the VPN Client and a VPN Gateway. The proper installation, configuration, and updating of the VPN Client are critical to its correct operation, so these objectives are included as well.

Appendix A.3 Security Problem Description Tables presents the Security Problem Description (SPD) in a more "traditional" form.

2.1 Unauthorized Access to User and TOE Data (T.UNAUTHORIZED_ACCESS)

This PP 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 client device. Therefore, the primary threat agents are the unauthorized entities that try to gain access to the protected network.

The gateway endpoint of the network communication can be both geographically and logically distant from the TOE, and 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 and/or manipulated directly by intermediate systems, leading to a compromise of the TOE. IPsec can be used to provide protection for this communication; however, there are a myriad of 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 remote VPN Gateway could be duped 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 VPN Gateway as if it were the TOE. In a similar manner, the TOE could also be duped into thinking that it is establishing communications with a legitimate remote VPN Gateway 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.

2.2 Inability to Configure the TSF (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 to configure certain aspects of the interface may also lead to the mis-specification of the desired communications policy or use of cryptography that may be desired or required for a particular users' site. This may result in unintended weak or plain-text 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.

2.3 Malicious Updates (T.UNAUTHORIZED_UPDATE)

Since the most common attack vector used involves attacking unpatched versions of software containing well-known flaws, updating the VPN client is necessary to ensure that changes to threat environment are addressed. Timely application of patches ensures that the client is a "hard target", thus increasing the likelihood that product will be able to maintain and enforce its security policy. However, the updates to be applied to the product must be trustable in some manner; otherwise, an attacker can write their own "update" that instead contains malicious code of their choosing, such as a rootkit, bot, or other malware. Once this "update" is installed, the attacker then has control of the system and all of its data.

Methods of countering this threat typically involve hashes of the updates, and potentially cryptographic operations (e.g., digital signatures) on those hashes as well. However, the validity of these methods introduces additional threats. For instance, a weak hash function could result in the attacker being able to modify the legitimate update in such a way that the hash remained unchanged. For cryptographic signature schemes, there are dependencies on

1) the strength of the cryptographic algorithm used to provide the signature, and
2) the ability of the end user to verify the signature (which typically involves checking a hierarchy of digital signatures back to a root of trust (a certificate authority)).

If a cryptographic signature scheme is weak, then it may be compromised by an attacker and the end user will install a malicious update, thinking that it is legitimate. Similarly, if the root of trust can be compromised, then a strong digital signature algorithm will not stop the malicious update from being installed (the attacker will just create their own signature on the update using the compromised root of trust, and the malicious update will then be installed without detection).

2.4 User Data Disclosure (T.USER_DATA_REUSE)

Data traversing the TOE could inadvertently be sent to a different user; 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.

2.5 TSF Failure (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 Security Objectives

The Security Objectives are the requirements for the Target of Evaluation (TOE) and for the Operational Environment derived from the threats, organizational security policies, and the assumptions in Section 2. Section 3 restates the Security Objectives for the TOE more formally as SFRs. The TOE is evaluated against the SFRs.

3.1 Establish VPN Tunnels

To address the issues concerning transmitting sensitive data between the TOE and the VPN Gateway described in Section 2.1, compliant TOEs will provide an encrypted channel for these communication paths between themselves and the VPN Gateway. These channels are implemented using IPsec. IPsec is specified by RFCs that offer a variety of implementation choices. Requirements have been imposed on some of these choices (particularly those for cryptographic primitives) to provide interoperability and resistance to cryptographic attack. While compliant TOEs must support all of the choices specified in the ST, they may support additional algorithms and protocols. If such additional mechanisms are not evaluated, guidance must be given to the administrator to make clear the fact that they are not evaluated.

In addition to providing protection from disclosure (and detection of modification) for the communications, IPsec offers two-way authentication of each endpoint in a cryptographically secure manner, meaning that even if there was a malicious attacker between the two endpoints, any attempt to represent themselves to either endpoint of the communications path as the other communicating party would be detected. This authentication is done using X.509 certificates to provide greater assurance in the authentication than is the case with pre-shared keys (although pre-shared keys may be supported by compliant TOEs as long as the use of certificates is also supported). The requirements on the IPsec protocol, in addition to the structure of the protocol itself, provides protection against replay attacks such as those described in Section 2.1.

(O.VPN_TUNNEL → FCS_CKM.1(1), FCS_CKM.1(2), FCS_CKM.2, FCS_CKM_EXT.4, FCS_COP.1(1), FCS_COP.1(2), FCS_COP.1(3), FCS_COP.1(4), FCS_IPSEC_EXT.1, FIA_X509_EXT.1, FTP_ITC.1)

3.2 Configuration of the TOE

To address the issues concerning the configuration of the TOE described in Section 2.2, the TOE will provide interfaces to control the configuration of IPsec and the underlying cryptographic mechanisms supporting the protocol, management of X.509 certificates, and updates to the TOE.

(O.TOE_CONFIGURATION → FMT_SMF.1)

3.3 Verifiable Updates

As outlined in Section 2.3, failure to verify that updates to the client can be trusted may lead to compromise of the security functionality. A first step in establishing trust in the update is to publish a hash of the update that can be verified prior to installing the update. While this establishes that the update downloaded is the one associated with the published hash, it does not indicate if the source of the update/hash combination has been compromised or can't be trusted. To establish trust in the source of the updates, a cryptographic mechanisms and procedures to procure the update, check the update cryptographically through the TOE-provided digital signature mechanism, and install the update will be provided.

(O.VERIFIABLE_UPDATES → FPT_TUD_EXT.1, FCS_COP.1(2), FCS_COP.1(3), FIA_X509_EXT.1)

3.4 Residual Information Clearing

In order to counter the threat that user data is inadvertently included in network traffic not intended by the original sender, the TSF ensures that network packets sent from the TOE do not include data "left over" from the processing of previous network information.

(O.RESIDUAL_INFORMATION_CLEARING → FDP_RIP.2)

3.5 TSF Self Test

In order to detect some number of failures of underlying security mechanisms used by the TSF, the TSF will perform self-tests. The extent of this self testing is left to the product developer, but a more comprehensive set of self tests should result in a more trustworthy platform on which to develop enterprise architecture.

(O.TSF_SELF_TEST → FPT_TST_EXT.1)

4 Security Requirements

The Security Functional Requirements included in this section are derived from Part 2 of the *Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4* (the CC), with additional extended functional components. The Security Assurance Requirements included in this section are derived from Part 3 of the CC. Supplemental Guidance is provided in the form of Assurance Activities associated with the functional requirements in Sections 4.1 and 4.2, as well as with the Security Assurance Requirements themselves in Section 4.3.

4.1 Security Functional Requirements for the VPN Client (TOE)

As indicated in Section 1.1.3.1, security functional requirements in the main body of the PP are divided into those that must be satisfied by the VPN client (the TOE), and those that must be satisfied by either the TOE or the platform on which it runs. This section contains the requirements that must be met by the TOE.

4.1.1 Class: Security Management (FMT)

The TOE is not required to maintain a separate management role. It is, however, required to provide functionality to configure certain aspects of TOE operation that should not be available to the general user population.

Specification of Management Functions (FMT_SMF)

FMT_SMF.1	**Specification of Management Functions**
FMT_SMF.1.1	The TOE shall be capable of performing the following management functions: • **Specify VPN gateways to use for connections,** • **Specify client credentials to be used for connections,** • **[assignment: any additional management functions].**

Application Note:

"Client credentials" will include the client certificate used for IPsec authentication, and may also include a username/password.

If there are additional management functions performed by the TOE (including those specified in Section 4.2.4, FMT_SMF), they should be added in the assignment.

Assurance Activity:

The evaluator shall check to ensure the TSS describes the client credentials and how they are used by the TOE.

The evaluator shall check to make sure that every management function mandated in the ST for this requirement are described in the operational guidance and that the description contains the information required to perform the management duties associated with each management function. The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE according to the operational guidance and testing each management activity listed in the Security Target.

Note that the testing here may be accomplished in conjunction with the testing of FCS_IPSEC_EXT.1.

4.2 Security Functional Requirements for the VPN Client or Client Platform

As indicated in Section 1.1.3.1, security functional requirements in the main body of the PP are divided into those that must be satisfied by the VPN client (the TOE), and those that must be satisfied by either the TOE or the platform on which it runs. This section contains requirements that must be met, but they can either be met by the TOE or the platform on which the TOE operates. In the case where the TOE relies on the platform, the platform must be evaluated either concurrently with or before the VPN Client. Assurance activities are therefore constructed such that those that apply when the requirements are met by the TOE are identified, and those that are performed when the platform on which the TOE operates implements the required functionality are likewise identified. If a test or documentation assurance activity is specified that is not specifically associated with either the TOE or the TOE platform, then it applies regardless of where the requirement is implemented.

4.2.1 Class: Cryptographic Support (FCS)

FCS_CKM.1(1) Cryptographic Key Generation (Asymmetric Keys)

FCS_CKM.1.1(1) **Refinement:** The [selection: TOE, TOE platform] shall generate **asymmetric** cryptographic keys **used for key establishment** in accordance with

- *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for finite field-based key establishment schemes;*
- *NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" for elliptic curve-based key establishment schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves] (as defined in FIPS PUB 186-4, "Digital Signature Standard")*
- *[selection: NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography" for RSA-based key establishment schemes, no other*]

and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

Application Note:
This component requires that the TOE/platform be able to generate the public/private key pairs that are used for key establishment purposes for the various cryptographic protocols used.

Since the domain parameters to be used are specified by the requirements of the protocol in this PP, it is not expected that the TOE/platform will generate domain parameters, and therefore there is no additional domain parameter validation needed when complying with the protocols specified in this PP.

Assurance Activity:

Requirement met by the platform
For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key establishment claimed in that platform's ST contains the key establishment requirement in the VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for

each supported platform) how the key establishment functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE
This assurance activity will verify the key generation and key establishments schemes used on the TOE.

Key Generation:
The evaluator shall verify the implementation of the key generation routines of the supported schemes using the applicable tests below.

Key Generation for RSA-Based Key Establishment Schemes
The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.
Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. *Random Primes:*
   - *Provable primes*
   - *Probable primes*
2. *Primes with Conditions:*
   - *Primes p1, p2, q1,q2, p and q shall all be provable primes*
   - *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*
   - *Primes p1, p2, q1,q2, p and q shall all be probable primes*

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Finite-Field Cryptography (FFC) – Based 56A Schemes
FFC Domain Parameter and Key Generation Tests
The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.
The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:
- *Primes q and p shall both be provable primes*
- *Primes q and field prime p shall both be probable primes*

and two ways to generate the cryptographic group generator g:
Cryptographic Group Generator:
- *Generator g constructed through a verifiable process*
- *Generator g constructed through an unverifiable process.*

The Key generation specifies 2 ways to generate the private key x:
Private Key:

- *len(q) bit output of RBG where 1 <=x <= q-1*
- *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1<= x<=q-1.*

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.
To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.
For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- *g != 0,1*
- *q divides p-1*
- *g^q mod p = 1*
- *g^x mod p = y*

for each FFC parameter set and key pair.

Key Generation for Elliptic Curve Cryptography (ECC)- Based 56A Schemes

ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

ECC Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Schemes

At this time, detailed test procedures for RSA-based key establishment schemes are not available. In order to show that the TSF complies with 800-56A and/or 800-56B, depending on the selections made, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the appropriate 800-56 standard(s) to which the TOE complies.*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the*

TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE.

For each applicable section of 800-56A and 800-56B (as selected), any omission of functionality related to "shall" or "should" statements shall be described.

FCS_CKM.1 (2) Cryptographic Key Generation (for asymmetric keys - IKE)

FCS_CKM.1.1(2) **Refinement**: The [selection: TOE, TOE platform] shall generate **asymmetric** cryptographic keys **used for IKE peer authentication** in accordance with a:

[selection:

- FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 for RSA schemes;
- FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4 for ECDSA schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curves];
- ANSI X9.31-1998, Appendix A.2.4 Using AES for RSA schemes]

and specified cryptographic key sizes *equivalent to, or greater than, a symmetric key strength of 112 bits*.

Application Note: The ANSI X9.31-1998 option will be removed from the selection in a future publication of this document. Presently, the selection is not exclusively limited to the FIPS PUB 186-4 options in order to allow industry some further time to complete the transition to the modern FIPS PUB 186-4 standard. As the preferred approach for cryptographic signature, elliptic curves will be required in future publications of this PP.

The keys that are required to be generated by the TOE through this requirement are intended to be used for the authentication of the VPN entities during the IKE (either v1 or v2) key exchange. While it is required that the public key be associated with an identity in an X509v3 certificate, this association is not required to be performed by the TOE, and instead is expected to be performed by a Certificate Authority in the Operational Environment.

As indicated in FCS_IPSEC_EXT.1, the TOE is required to implement support for RSA or ECDSA (or both) for authentication.

See NIST Special Publication 800-57, "Recommendation for Key Management" for information about equivalent key strengths.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the key generation function claimed in that platform's ST contains the key generation requirement in the

VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) how the key generation functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

If the TSF implements a FIPS 186-4 signature scheme, this requirement is verified under FCS_COP.1(2).

If the ESF implements the ANSI X9.31-1998 scheme, the evaluator shall check to ensure that the TSS describes how the key-pairs are generated. In order to show that the TSF implementation complies with ANSI X9.31-1998, the evaluator shall ensure that the TSS contains the following information:

- *The TSS shall list all sections of the standard to which the TOE complies;*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;*
- *For each applicable section of Appendix B, any omission of functionality related to "shall" or "should" statements shall be described.*

FCS_CKM_EXT.2 Cryptographic Key Storage

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

Application Note:

This requirement ensures that persistent secrets (credentials, secret keys) and private keys are stored securely when not in use. If some secrets/keys are manipulated by the TOE and others are manipulated by the platform, then both of the selections can be specified by the ST author.

Assurance Activity:

Regardless of whether this requirement is met by the TOE or the TOE platform, the evaluator will check the TSS to ensure that it lists each persistent secret (credential, secret key) and private key needed to meet the requirements in the ST. For each of these items, the evaluator will confirm that the TSS lists for what purpose it is used, and how it is stored. The evaluator than performs the following actions.

Persistent secrets and private keys manipulated by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the persistent secrets and private keys listed as being stored by the platform in the VPN client ST are identified as being protected in that platform's ST.

Persistent secrets and private keys manipulated by the TOE

The evaluator reviews the TSS for to determine that it makes a case that, for each item listed as being manipulated by the TOE, it is not written unencrypted to persistent memory, and that the item is stored by the platform.

FCS_CKM_EXT.4	**Cryptographic Key Zeroization**
FCS_CKM_EXT.4.1	**Refinement:** The [selection: TOE, TOE platform] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

Application Note:

Any security related information (such as keys, authentication data, and passwords) must be zeroized when no longer in use to prevent the disclosure or modification of security critical data.

The zeroization indicated above applies to each intermediate storage area for plaintext key/CSP (i.e., any storage, such as memory buffers, that is included in the path of such data) upon the transfer of the key/CSP to another location.

In practice, the TOE will not implement all of the functionality associated with the requirement, since if it performs zeroization at all it will be by invoking platform interfaces to perform the storage location clear/overwrite function. The ST author should select "TOE" when, for at least one of the keys needed to meet the requirements of this PP, the TOE manipulates (reads, writes) the data identified in the requirement and thus needs to ensure that those data are cleared. In these cases,, it is sufficient for the TOE to invoke the correct underlying functions of the host to perform the zeroization--it does not imply that the TOE has to include a kernel-mode memory driver to ensure the data are zeroized.

In the likely event that some of the data are manipulated by the TOE and other data are manipulated entirely by the platform, the ST author shall select both options and make it clear in the TSS the entity responsible (TOE, TOE platform) for performing the zeroization.

Assurance Activity:

The evaluator shall ensure that all plaintext secret and private cryptographic keys and CSPs (whether manipulated by the TOE or exclusively by the platform) are identified in the VPN Client ST's TSS, and that they are accounted for by the assurance activities in this section.

Requirement met by the platform

The evaluator shall check to ensure the TSS describes each of the secret keys (keys used for symmetric encryption), private keys, and CSPs used to generate key that are not otherwise covered by the FCS_CKM_EXT.4 requirement levied on the TOE.

For each platform listed in the ST, the evaluator shall examine the TSS of the ST of the platform to ensure that each of the secret keys, private keys, and CSPs used to generate key listed above are covered.

Requirement met by the TOE

The evaluator shall check to ensure the TSS describes when each of the plaintext keys are cleared (e.g., system power off, disconnection of an IPsec connection, when no longer needed by the VPN channel per the protoco); and the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase). If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

For each key clearing situation described in the TSS, the evaluator shall repeat the following test.

Test 1: The evaluator shall utilize appropriate combinations of specialized operational environment and development tools (debuggers, simulators, etc.) for the TOE and instrumented TOE builds to test that keys are cleared correctly, including all intermediate copies of the key that may have been created internally by the TOE during normal cryptographic processing with that key.

Cryptographic TOE implementations in software shall be loaded and exercised under a debugger to perform such tests. The evaluator shall perform the following test for each key subject to clearing, including intermediate copies of keys that are persisted encrypted by the TOE:

1. *Load the instrumented TOE build in a debugger.*
2. *Record the value of the key in the TOE subject to clearing.*
3. *Cause the TOE to perform a normal cryptographic processing with the key from #1.*
4. *Cause the TOE to clear the key.*
5. *Cause the TOE to stop the execution but not exit.*
6. *Cause the TOE to dump the entire memory footprint of the TOE into a binary file.*
7. *Search the content of the binary file created in #4 for instances of the known key value from #1.*

The test succeeds if no copies of the key from #1 are found in step #7 above and fails otherwise.

The evaluator shall perform this test on all keys, including those persisted in encrypted form, to ensure intermediate copies are cleared.

Cryptographic Operation (FCS_COP)

FCS_COP.1(1)	**Cryptographic Operation (Data Encryption/Decryption)**
FCS_COP.1.1(1)	**Refinement:** The [selection: TOE, TOE platform] shall perform [*encryption and decryption*] in accordance with a specified cryptographic algorithm *AES operating in* **GCM and CBC mode** with cryptographic key sizes 128-bits and 256-bits that meets the following: • **FIPS PUB 197, "Advanced Encryption Standard (AES)"** • **NIST SP 800-38D, NIST SP 800-38A.**

Application Note:

This PP requires the modes GCM and CBC to be used in the IPsec and IKE protocols (FCS_IPSEC_EXT.1.4, FCS_IPSEC_EXT.1.6). Therefore, the FCS_COP.1.1(1) element in the NDPP has been specified here to ensure the ST Author includes these two modes to be consistent with the IPsec requirements.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the encryption/decryption function(s) claimed in that platform's ST contains the encryption/decryption function(s) in the VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) how the encryption/decryption functionality is invoked for the indicated modes and key sizes in the VPN Client's ST (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

The evaluator shall perform the following activities based on the selections in the ST.

AES-CBC Tests

AES-CBC Known Answer Tests
There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

> ***KAT-1.*** *To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.*
>
> *To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.*
>
> ***KAT-2.*** *To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.*
>
> *To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.*

KAT-3. *To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].*

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. *To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].*

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1< i <=10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
        if i == 1:
                CT[1] = AES-CBC-Encrypt(Key, IV, PT)
                PT = IV
```

else:

CT[i] = AES-CBC-Encrypt(Key, PT)
PT = CT[i-1]

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Monte Carlo Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

***Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.*

***Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.*

***Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.*

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

FCS_COP.1(2) Cryptographic Operation (for cryptographic signature)

FCS_COP.1.1(2) **Refinement:** The [selection: TOE, TOE platform] shall perform **cryptographic signature services** in accordance with a specified cryptographic algorithm:

- **[selection: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 for RSA scheme**
- **FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4 for ECDSA schemes and implementing "NIST curves" P-256, P-384 and [selection: P-521, no other curve]]**

and cryptographic key sizes [***equivalent to, or greater than, a symmetric key strength of 112 bits***]**.**

Application Note:

The ST Author should choose the algorithm implemented to perform digital signatures; if more than one algorithm is available, this requirement should be iterated to specify the functionality. For the algorithm chosen, the ST author should make the appropriate assignments/selections to specify the parameters that are implemented for that algorithm.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the digital signature functions claimed in that platform's ST contains the digital signature functions in the VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) how the digital signature functionality is invoked for each operation they are used for in the VPN client (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

The evaluator shall perform the following activities based on the selections in the ST.

Key Generation:

Key Generation for RSA Signature Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. *Random Primes:*
   - *Provable primes*
   - *Probable primes*
2. *Primes with Conditions:*
   - *Primes p1, p2, q1,q2, p and q shall all be provable primes*
   - *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*

- *Primes p1, p2, q1,q2, p and q shall all be probable primes*

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

ECDSA Key Generation Tests

FIPS 186-4 ECDSA Key Generation Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-284 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

ECDSA Algorithm Tests

CDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-284 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-284 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

FCS_COP.1(3)	**Cryptographic Operation (Cryptographic Hashing)**
FCS_COP.1.1(3)	**Refinement:** The [selection: TOE, TOE platform] shall perform [*cryptographic hashing services*] in accordance with a specified cryptographic algorithm [**selection: SHA-1, SHA-256, SHA-384, SHA-512] and message digest sizes** [**selection: 160, 256, 384, 512**] **bits** that meet the following: *FIPS Pub 180-4, "Secure Hash Standard."*

Application Note:
In future versions of this document, SHA-1 may be removed as an option. SHA-1 for generating digital signatures will no longer be allowed after December 2013, and SHA-1 for verification of digital signatures is strongly discouraged as there may be risk in accepting these signatures.

The selection of the hashing algorithm must correspond to the selection of the message digest size; for example, if SHA-1 is chosen, then the only valid message digest size selection would be 160 bits.

Assurance Activity:

The evaluator shall check that the association of the hash function with other cryptographic functions (for example, the digital signature verification function) specified in the VPN Client ST (whether these are performed by the platform or by the TOE) is documented in the TSS.

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the hash function(s) claimed in that platform's ST contains the hash function(s) in the VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) how the hash functionality is invoked for each digest size selected in the VPN Client's ST (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode
The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode
The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode
*The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Selected Long Messages Test - Byte-oriented Mode
*The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

Pseudorandomly Generated Messages Test
This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

FCS_COP.1(4)	**Cryptographic Operation (Keyed-Hash Message Authentication)**
FCS_COP.1.1(4)	**Refinement:** The [selection: TOE, TOE platform] shall perform **keyed-hash message authentication** in accordance with a specified cryptographic algorithm **HMAC-** [selection: SHA-1, SHA-256, SHA-384, SHA-512],-key size **[assignment: key size (in bits) used in HMAC], and message digest size of** [selection: 160, 256, 384, 512] **bits** that meet the following: **FIPS PUB 198-1, "The Keyed-Hash Message Authentication Code", and FIPS PUB 180-4, "Secure Hash Standard"**.

Application Note:
The selection of the hashing algorithm must correspond to the selection of the message digest size; for example, if HMAC-SHA-256 is chosen, then the only valid message digest size selection would be 256 bits.

The message digest size above corresponds to the underlying hash algorithm used. Note that truncating the output of the HMAC following the hash calculation is an appropriate step in a variety of applications. This does not invalidate compliance with this requirement, however, the ST should state that truncation is performed, the size of the final output, and the standard to which this truncation complies.

Assurance Activity:

The evaluator shall check that the association of the keyed-hash function with other cryptographic functions specified in the VPN Client ST (whether these are performed by the platform or by the TOE) is documented in the TSS.

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the keyed hash function(s) claimed in that platform's ST contains the keyed hash function(s) in the VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) how the keyed hash functionality is invoked for each digest size and key size selected in the VPN Client's ST (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

Additionally, for all cases where the output of the HMAC following the hash calculation is truncated, the evaluator shall ensure that the TSS states for what operation this truncation takes place; the size of the final output; and the standard to which this truncation complies.

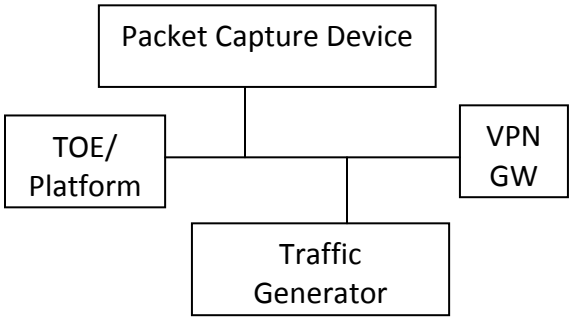
The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key-length, hash function used, block size, and output MAC length used.

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known good implementation.

Extended: Internet Protocol Security (FCS_IPSEC_EXT)

In order to show that the TSF implements the RFCs in accordance with the requirements of this PP, the evaluator shall perform the assurance activities listed below. In future versions of this PP, assurance activities may be augmented, or new ones introduced that cover more aspects of RFC compliance than are currently described in this publication.

The TOE is required to use the IPsec protocol to establish connections used to communicate with a VPN Gateway.



The evaluators shall minimally create a test environment equivalent to the test environment illustrated above. It is expected that the traffic generator is used to construct network packets and will provide the evaluator with the ability manipulate fields in the ICMP, IPv4, IPv6, UDP, and TCP packet headers. The evaluators must provide justification for any differences in the test environment.

In the following elements of the FCS_IPSEC_EXT.1 component, it is allowable for some or all of the individual elements to be implemented by the platform on which the VPN client operates. The TSS of the VPN client will identify all of the information listed in the assurance activities of the elements; this may be repeated from the underlying platform's (if implemented by the platform) ST and should be indicated as such in the VPN client ST. If the configuration is to be performed on the platform, the evaluators shall ensure that the "operational guidance" for each platform in the VPN Client ST contains the appropriate information (either through reference in the platform's ST, or by information contained in the VPN client ST). All tests must be performed by the evaluators using the VPN client and a representative sample of platforms listed in the VPN Client ST.

FCS_IPSEC_EXT.1

Extended: Internet Protocol Security (IPsec) Communications

FCS_IPSEC_EXT.1.1 The [selection: TOE, TOE platform] shall implement the IPsec architecture as specified in RFC 4301.

Assurance Activity:

The evaluator shall examine the operational guidance to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for DISCARD, BYPASS and PROTECT.
The evaluator uses the operational guidance to configure the TOE and platform to carry out the following tests:
Test 1: The evaluator shall configure the SPD such that there is a rule for DISCARD, BYPASS, PROTECT. The selectors used in the construction of the rule shall be different such that the evaluator can send in three network packets with the appropriate fields in the packet header that each packet will match one of the three rules. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packet was dropped, allowed through without modification, was encrypted by the IPsec implementation.

Test 2: The evaluator shall devise two equal SPD entries with alternate operations – BYPASS and PROTECT. The entries should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first entry is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.
Test 3: The evaluator shall repeat the procedure above, except that the two entries should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

FCS_IPSEC_EXT.1.2 The [selection: TOE, TOE platform] shall implement [selection: tunnel mode, transport mode].

Assurance Activity:

The evaluator checks the TSS to ensure it states that the VPN can be established to operate in tunnel mode and/or transport mode (as selected). The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

The evaluator shall perform the following test(s) based on the selections chosen:

Test 1 (conditional): If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE/platform to operate in tunnel mode and also configures a VPN GW to operate in tunnel mode. The evaluator configures the TOE/platform and the VPN GW to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the client to connect to the VPN GW peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

Test 2 (conditional): If transport mode is selected, the evaluator uses the operational guidance to configure the TOE/platform to operate in transport mode and also configures a VPN GW to operate in transport mode. The evaluator configures the TOE/platform and the VPN GW to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE/platform to connect to the VPN GW. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

FCS_IPSEC_EXT.1.3 The [selection: TOE, TOE platform] shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

Assurance Activity:

The evaluator shall examine the TSS to verify that the TSS provides a description of how a packet is processed against the SPD and that if no "rules" are found to match, that a final rule exists, either implicitly or explicitly, that causes the network packet to be discarded.

The evaluator checks that the operational guidance provides instructions on how to construct the SPD and uses the guidance to configure the TOE/platform for the following tests.

The evaluator shall perform the following test:

Test 1: The evaluator shall configure the SPD such that it has entries that contain operations that DISCARD, BYPASS, and PROTECT network packets. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches a BYPASS entry and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE/platform created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was not permitted to flow to any of the TOE's interfaces.

FCS_IPSEC_EXT.1.4 The [selection: TOE, TOE platform] 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, [selection: AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC, no other algorithms].

Assurance Activity:

The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4) Cryptographic Operations (for keyed-hash message authentication).

The evaluator checks the operational guidance to ensure it provides instructions on how to configure the TOE/platform to use the AES-GCM-128, and AES-GCM-256 algorithms, and if either AES-CBC-128 or AES-CBC-256 have been selected the guidance instructs how to use these as well.

Test 1: The evaluator shall configure the TOE/platform as indicated in the operational guidance configuring the TOE/platform to using each of the AES-GCM-128, and AES-GCM-256 algorithms, and attempt to establish a connection using ESP. If the ST Author has selected either AES-CBC-128 or AES-CBC-256, the TOE/platofrm is configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.

FCS_IPSEC_EXT.1.5 The [selection: TOE, TOE platform] shall implement the protocol: [selection: IKEv1 as defined in RFCs 2407, 2408, 2409, RFC 4109, [selection: no other RFCs for extended sequence numbers, RFC 4304 for extended sequence numbers], and [selection: no other RFCs for hash functions, RFC 4868 for hash functions]; IKEv2 as defined in RFCs 5996 (with mandatory support for NAT traversal as specified in section 2.23), 4307, and [selection: no other RFCs for hash functions, RFC 4868 for hash functions]].

Assurance Activity:

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

The evaluator shall check the operational guidance to ensure it instructs the administrator how to configure the TOE/platform to use IKEv1 and/or IKEv2 (as selected), and uses the guidance to configure the TOE/platform to perform NAT traversal for the following test.

Test 1: The evaluator shall configure the TOE/platform so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

FCS_IPSEC_EXT.1.6 The [selection: TOE, TOE platform] shall ensure the encrypted payload in the [selection: IKEv1, IKEv2] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [selection: AES-GCM-128, AES-GCM-256 as specified in RFC 5282, no other algorithm].

Assurance Activity:

The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms AES-CBC-128, AES-CBC-256 are specified, and if others are chosen in the selection of the requirement, those are included in the TSS discussion.

The evaluator ensures that the operational guidance describes the configuration of the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE/platform to perform the following test for each ciphersuite selected.

Test 1: The evaluator shall configure the TOE/platform to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

FCS_IPSEC_EXT.1.7 The [selection: TOE, TOE platform] shall ensure that IKEv1 Phase 1 exchanges use only main mode.

Assurance Activity:

The evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

If the mode requires configuration of the TOE/platform prior to its operation, the evaluator shall check the operational guidance to ensure that instructions for this configuration are contained within that guidance.

Test 1 (conditional): The evaluator shall configure the TOE/platform as indicated in the operational guidance, and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported. This test is not applicable if IKEv1 is not selected above in the FCS_IPSEC_EXT.1.5 protocol selection.

FCS_IPSEC_EXT.1.8 The [selection: TOE, TOE platform] shall ensure that [selection: IKEv2 SA lifetimes can be configured by [selection: an Administrator, VPN Gateway] based on [selection: number of packets/number of bytes; length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs], IKEv1 SA lifetimes can be configured by an [selection: an Administrator, VPN Gateway] based on [selection: number of packets/number of bytes ; length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs]].

Application Note:

The ST Author is afforded a selection based on the version of IKE in their implementation. There is a further selection within this selection that allows the ST Author to specify which entity is responsible for "configuring" the life of the SA. An implementation that allows an administrator to configure the client or a VPN gateway that pushes the SA lifetime down to the client are both acceptable.

As far as SA lifetimes are concerned, the TOE can limit the lifetime based on the number of bytes transmitted, or the number of packets transmitted. Either packet-based or volume-based SA lifetimes are acceptable; the ST author makes the appropriate selection to indicate which type of lifetime limits are supported.

Assurance Activity:

The evaluator verifies that the values for SA lifetimes can be configured and that the instructions for doing so are located in the operational guidance. If time-based limits are supported, the evaluator ensures that either the Administrator or VPN Gateway are able to configurable Phase 1 SAs values for 24 hours and 8 hours for Phase 2 SAs. Currently there are no values mandated for the number of packets or number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered." Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

Test 1 (Conditional): The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.
Test 2 (Conditional): The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.
Test 3 (Conditional): The evaluator shall perform a test similar to Test 1 for Phase 2 SAs, except that the lifetime will be 8 hours instead of 24.

FCS_IPSEC_EXT.1.9 The [selection: TOE, TOE platform] shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [assignment: (one or more) number(s) of bits that is at least twice the "bits of security" value associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, *Recommendation for Key Management – Part 1: General*] bits.

Assurance Activity:
The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" and the nonces meet the stipulations in the requirement.

FCS_IPSEC_EXT.1.10 The [selection: TOE, TOE platform] 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^[assignment: (one or more) "bits of security" value(s) associated with the negotiated Diffie-Hellman group as listed in Table 2 of NIST SP 800-57, *Recommendation for Key Management – Part 1: General*] .

Assurance Activity:
The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x" (as defined in FCS_IPSEC_EXT.1.9) and each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" and the nonces meet the stipulations in the requirement.

FCS_IPSEC_EXT.1.11 The [selection: TOE, TOE platform] shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), and [selection: 5 (1536-bit MODP), 24 (2048-bit MODP with 256-bit POS), 20 (384-bit Random ECP), [assignment: other DH groups that are implemented by the TOE], no other DH groups].

Assurance Activity:

The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer. The evaluator shall also perform the following test:

Test 1: For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

FCS_IPSEC_EXT.1.12 The [selection: TOE, TOE platform] shall ensure that all IKE protocols perform peer authentication using a [selection: *RSA, ECDSA*] that use X.509v3 certificates that conform to RFC 4945 and [selection: Pre-shared Keys, no other method].

Assurance Activity:

The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(2) Cryptographic Operations (for cryptographic signature).

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established. The description in the TSS and the operational guidance shall also indicate how pre-shared key establishment is accomplished for TOEs/platforms that can generate a pre-shared key as well as TOEs/platforms that simply use a pre-shared key.

The evaluator ensures the operational guidance describes how to set up the TOE/platform to use the cryptographic algorithms RSA and/or ECDSA.

In order to construct the environment and configure the TOE/platform for the following tests, the evaluator will ensure that the operational guidance also describes how to configure the TOE/platform to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE/platform and marked "trusted".

*For efficiency sake, the testing that is performed here has been combined with the testing for **FIA_X509_EXT.2.1** (for IPsec connections), **FCS_IPSEC_EXT.1.13**, and **FIA_X509_EXT.2.3**. The following tests shall be repeated for each peer authentication protocol selected in the **FCS_IPSEC_EXT.1.12** selection above:*

Test 1: The evaluator shall have the TOE/platform generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE/platform and the peer VPN used to establish a connection) for its signature. The values for the DN (Common Name, Organization, Organizational Unit, and Country) will also be passed in the request.

Test 2: The evaluator shall use a certificate signed using the RSA or ECDSA algorithm to authenticate the remote peer during the IKE exchange. This test ensures the remote peer has the certificate for the trusted CA that signed the TOE's certificate and it will do a bit-wise comparison on the DN. This bit-wise comparison of the DN ensures that not only does the peer have a certificate signed by the trusted CA, but the certificate is from the DN that is expected. The evaluator will configure the TOE/platform to associate a certificate (e.g., a certificate map in some implementations) with a VPN connection. This is what the DN is checked against.

Test 3: The evaluator shall test that the TOE/platform can properly handle revoked certificates – conditional on whether CRL or OCSP is selected; if both are selected, and then a test is performed for each method. For this draft of the PP, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the SA is established. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the TOE/platform will not establish an SA.

Test 4: The evaluator shall test that given a signed certificate from a trusted CA, that when the DN does not match – any of the four fields can be modified such that they do not match the expected value, that an SA does not get established.

Test 5: The evaluator shall ensure that the TOE is configurable to either establish an SA, or not establish an SA if a connection to the certificate validation entity cannot be reached. For each method selected for certificate validation, the evaluator attempts to validate the certificate – for the purposes of this test, it does not matter if the certificate is revoked or not. For the "mode" where an SA is allowed to be established, the connection is made. Where the SA is not to be established, the connection is refused.
Test 6 [conditional]: The evaluator shall generate a pre-shared key and use it, as indicated in the operational guidance, to establish an IPsec connection with the VPN GW peer. If the generation of the pre-shared key is supported, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE/platform generating the key as well as an instance of the TOE/platform merely taking in and using the key.

FCS_IPSEC_EXT.1.13 The [selection: TOE, TOE platform] shall not establish an SA if the distinguished name (DN) contained in a certificate does not match the expected DN for the entity attempting to establish a connection.

Assurance Activity:

Assurance activities for this element are tested through assurance activities for ***FCS_IPSEC_EXT.1.12.***

FCS_IPSEC_EXT.1.14 The [selection: TOE, TOE platform] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: *IKEv1 Phase 1, IKEv2 IKE_SA*] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [selection: *IKEv1 Phase 2, IKEv2 CHILD_SA*] connection.

Assurance Activity:

The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

The evaluator simply follows the guidance to configure the TOE/platform to perform the following tests.

Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

Assurance Activity:

The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.
Application Notes:

FCS_IPSEC_EXT.1.7 is only applicable if IKEv1 is selected.

FCS_IPSEC_EXT.1.8: The ST author chooses either the IKEv1 requirements or IKEv2 requirements (or both, depending on the selection in FCS_IPSEC_EXT.1.5. The IKEv1 requirement can be accomplished either by providing Authorized Administrator-configurable lifetimes (with appropriate instructions in documents mandated by AGD_OPE), or by "hard coding" the limits in the implementation. For IKEv2, there are no hardcoded limits, but in this case it is required than an administrator be able to configure the values. In general, instructions for setting the parameters of the implementation, including lifetime of the SAs, should be included in the operational guidance generated for AGD_OPE. It is appropriate to refine the requirement in terms of number of MB/KB instead of number of packets, as long as the TOE is capable of setting a limit on the amount of traffic that is protected by the same key (the total volume of all IPsec traffic protected by that key). It is also appropriate to refine the requirement such that SA lifetime management and enforcement occurs external to the TOE (i.e.: on a VPN Gateway), however, even when the requirement is refined in this manner, the evaluator shall conduct the associated assurance activities described above.

Since the implementation may allow different Diffie-Hellman groups to be negotiated for use in forming the SAs, the assignments in FCS_IPSEC_EXT.1.9 and FCS_IPSEC_EXT.1.10 may contain multiple values. For each DH group supported, the ST author consults Table 2 in 800-57 to determine the "bits of security" associated with the DH group. Each unique value is then used to fill in the assignment (for 1.9 they are doubled; for 1.10 they are inserted directly into the assignment). For example, suppose the implementation supports DH group 14 (2048-bit MODP) and group 20 (ECDH using NIST curve P-384). From Table 2, the bits of security value for group 14 is 112, and for group 20 it is 192. For FCS_IPSEC_EXT.1.9, then, the assignment would read "[224, 384]" and for FCS_IPSEC_EXT.1.10 it would read "[112,192]" (although in this case the requirement should probably be refined so that it makes sense mathematically).

FCS_IPSEC_EXT.1.11: The selection is used to specify additional DH groups supported. This applies to IKEv1 and IKEv2 exchanges. In future versions of this PP, DH Group 20 (384-bit RandomECP) will be required. It should be noted that if any additional DH groups are specified, they must comply with the requirements (in terms of the ephemeral keys that are established) listed in FCS_CKM.1.

FCS_IPSEC_EXT.1.12: At least one public-key-based Peer Authentication method is required in order to conform to this PP; one or more of the public key schemes is chosen by the ST author to reflect what is implemented. The ST author also ensures that appropriate FCS requirements reflecting the algorithms used (and key generation capabilities, if provided) are listed to support those methods. Note that the TSS will elaborate on the way in which these algorithms are to be used (for example, 2409 specifies three authentication methods using public keys; each one supported will be described in the TSS).

FCS_IPSEC_EXT.1.14: The ST author chooses either or both of the IKE selections based on what is implemented by the TOE. Obviously, the IKE version(s) chosen should be consistent not only in this element, but with other choices for other elements in this component. While it is acceptable for this capability to be configurable, the default configuration in the evaluated configuration (either "out of the box" or by configuration guidance in the AGD documentation) must enable this functionality.

Extended: Cryptographic Operation (Random Bit Generation) (FCS_RBG_EXT)

FCS_RBG_EXT.1 Extended: Cryptographic operation (Random Bit Generation)

FCS_RBG_EXT.1.1 The [selection: TOE, TOE platform] shall perform all deterministic random bit generation services in accordance with [selection, *choose one of: NIST Special Publication 800-90A using* [selection: *Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES), Dual_EC_DRBG (any)]; FIPS Pub 140-2 Annex C: X9.31 Appendix 2.4 using AES*].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [selection: *a software-based noise source, a platform-based RBG*] with a minimum of [selection: *128 bits, 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.

Application Note:
NIST Special Pub 800-90B, Appendix C describes the minimum entropy measurement that should be used immediately and will be required in future versions of this PP.

For the first selection in FCS_RBG_EXT.1.2, the ST author indicates whether the sources of entropy are software-based or platform-based, or both. If there are multiple sources of entropy, the ST will describe each entropy source and whether it is software- or platform-based. Platform-based noise sources are preferred.

The platform-based RBG source is the output of a validated RBG provided by the platform, which is used as an entropy source for a TSF-provided DRBG according to FCS_RBG_EXT.1.1. In this way, the developer has chained RBGs as described in NIST SP800-90C.

For the second and third selections in FCS_RBG_EXT.1.1, the ST author should select the standard to which the RBG services comply (either 800-90A or 140-2 Annex C).

For the selection in FCS_RBG_EXT.1.2, the ST author selects the appropriate number of bits of entropy that corresponds to the greatest security strength of the algorithms included in the ST. Security strength is defined in Tables 2 and 3 of NIST SP 800-57A. For example, if the implementation includes 2048-bit RSA (security strength of 112 bits), AES 128 (security strength 128 bits), and HMAC-512 (security strength 256 bits), then the ST author would select 256 bits.

SP 800-90A contains four different methods of generating random numbers; each of these, in turn, depends on underlying cryptographic primitives (hash functions/ciphers). The ST author will select the function used (if 800-90A is selected), and include the specific underlying cryptographic primitives used in the requirement or in the TSS. While any of the identified hash functions (SHA-1, SHA-224, SHA-256, SHA-384, SHA-512) are allowed for Hash_DRBG or HMAC_DRBG, only AES-based implementations for CT_DRBG are allowed. While any of the curves defined in 800-90A are allowed for Dual_EC_DRBG, the ST author not only must include the curve chosen, but also the hash algorithm used.

Note that for FIPS Pub 140-2 Annex C, currently only the method described in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3 is valid. If the key length for the AES implementation used here is different than that used to encrypt the user data, then FCS_COP.1 may have to be adjusted or iterated to reflect the different key length. For the selection in FCS_RBG_EXT.1.2, the ST author selects the minimum number of bits of entropy that is used to seed the RBG.

The ST author also ensures that any underlying functions are included in the baseline requirements in the ST (if these are implemented by the platform, then the appropriate selections are made as well).

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the RBG functions claimed in that platform's ST contains the RBG functions in the VPN Client's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) how the RBG functionality is invoked for each operation they are used for in the VPN client (it should be noted that this may be through a mechanism that is not implemented by the VPN Client; nonetheless, that mechanism will be identified in the TSS as part of this assurance activity).

Requirement met by the TOE

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix E.

If the ST author has selected a platform-based noise source, the evaluator shall verify that the platform's RBG has been validated by examining the platform's ST. The evaluator shall verify that the platform's RBG is seeded with at least the amount of entropy selected by the ST author for this profile. In this case, the ST author is not responsible for Annex E documentation of the platform's RBG.

The evaluator shall also perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to FIPS 140-2, Annex C

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length

appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.

The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section 3. The evaluators ensure that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate drbg, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "Generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate drbg, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

***Entropy input:** the length of the entropy input value must equal the seed length.*
***Nonce:** If a nonce is supported (CTR_DRBG with no df does not use a nonce), the nonce bit length is one-half the seed length.*
***Personalization string:** The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.*
***Additional input:** the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.*

4.2.2 Class: User Data Protection (FDP)

Residual Information Protection (FDP_RIP)

FDP_RIP.2	Full Residual Information Protection
FDP_RIP.2.1	The [selection: TOE, TOE platform] shall enforce that any previous information content of a resource is made unavailable upon the [selection: *allocation of the resource to*, *deallocation of the resource from*] all objects.

Application Note:

This requirement ensures, for example, that protocol data units (PDUs) are not padded with residual information such as cryptographic key material. The ST author uses the selection to specify when previous information is made unavailable.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that residual information protection measures with respect to network packets passing through the platform are claimed in that platform's ST. The evaluator shall also examine the TSS of the VPN Client's ST to verify that it describes (for each supported platform) the extent to which the client processes network packets and addresses the FDP_RIP.2 requirement.

Requirement met by the TOE

"Resources" in the context of this requirement are network packets being sent through (as opposed to "to", as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.

4.2.3 Class: Identification and Authentication (FIA)

The baseline requirements for the TOE are fairly limited with respect to I&A, since no formal administrative or general purpose users are defined. The extent of the I&A required to be performed by the TOE relates to the authentication done at the machine level when establishing the IPsec connection. These I&A requirements are specified in the FCS_IPSEC_EXT.1 component to keep requirements on the IPsec protocol grouped together for understandability as well as for ease of authoring and applying assurance activities. While the requirements on certificates used in the IPsec peer authentication exchanges are listed here, some of the assurance activities involving certificates are specified in the FCS_IPSEC_EXT.1 assurance activities.

X509 Certificates (FIA_X509_EXT)

FIA_X509_EXT.1 Extended: X.509 Certificate Validation

FIA_X509_EXT.1.1 The [selection: TOE, TOE platform] shall validate certificates in accordance with the following rules:

- Perform RFC 5280 certificate validation and certificate path validation.
- Validate the revocation status of the certificate using [selection: the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759].
- Validate the certificate path by ensuring the basicConstraints extension is present and the cA flag is set to TRUE for all CA certificates.
- Validate the extendedKeyUsage field according to the following rules:
    - Certificates used for [selection: trusted updates, integrity verification, no other purpose] shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).

Application Note:

FIA_X509_EXT.1.1 *lists the rules for validating certificates. The ST author shall select whether revocation status is verified using OCSP or CRLs. Certificates are used for IPsec peer authentication (**FCS_IPSEC_EXT.1**), and optionally for trusted updates of TSF software (**FPT_TUD_EXT.1**) and for integrity verification (**FPT_TST_EXT.1.2**) and, if implemented, must be validated to contain the Code Signing purpose extendedKeyUsage.*

It should be noted that the validation is expected to end in a trusted root certificate.

FIA_X509_EXT.1.2 The [selection: TOE, TOE platform] shall only treat a certificate as a CA certificate if the following is met: the basicConstraints extension is present and the CA flag is set to TRUE.

Assurance Activity:

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place – the TOE or the TOE platform. It may be that the TOE requests the platform to perform the check and provide a result, or the TOE may do the check itself. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm, ensuring that it describes how the validation chain will terminate in a trusted root certificate.

The evaluator ensures the guidance documentation provides the user with the necessary information to setup the validation check whether it is done by the TOE or TOE platform. The guidance documentation provides instructions how to select the method used for checking, as well as how to setup a protected communication path with the entity providing the information pertaining to certificate validity.

*Regardless of the selection of "TOE" or "TOE Platform in the **FIA_X509_EXT.1** elements, the evaluator shall perform the following tests. This testing may be combined with the testing performed in the other assurance activities (e.g., for **FCS_IPSEC_EXT.1.12**).*

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function (trusted channel setup, trusted software update, integrity check) failing. The evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

Test 3: The evaluator shall test that revoked certificates are properly handled – conditional on whether CRL or OCSP is selected; if both are selected, and then a test is performed for each method. For this draft of the PP, the evaluator has to only test one up in the trust chain (future drafts may require to ensure the validation is done up the entire chain). The evaluator shall ensure that a valid certificate is used, and that the SA is established. The evaluator then attempts the test with a certificate that will be revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that an SA will not be established.

Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the certificate does not contain the basicConstraints extension. The validation of the certificate path fails.

Test 5: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.
Test 6: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

FIA_X509_EXT.2 Extended: X.509 Certificate Use and Management

FIA_X509_EXT.2.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec exchanges, and [selection: *digital signatures for FPT_TUD_EXT.1, integrity checks for FPT_TST_EXT.1.2, no additional uses*].

Application Note:

If certificate-based digital signatures are used for the code integrity checks as per FPT_TST_EXT.1, the ST author will make this selection; otherwise select "no additional uses".

Assurance Activity:

Assurance activities for this element are tested through assurance activities for FCS_IPSEC_EXT.1, (conditionally) FPT_TUD_EXT.1, and (conditionally) FPT_TST_EXT.1.

FIA_X509_EXT.2.2 When a connection to determine the validity of a certificate cannot be established, the [selection: TOE, TOE platform] shall [selection: *allow the administrator to choose whether to accept the certificate in these cases, accept the certificate, not accept the certificate*].

Application Note:

Often a connection must be established to perform a verification of the revocation status of a certificate - either to download a CRL or to perform OCSP. The selection is used to describe the behavior in the event that such a connection cannot be established (for example, due to a network error). The behavior of the TOE in these cases is described by the second selection. If the TOE has determined the certificate valid according to all other rules in FIA_X509_EXT.1, the behavior indicated in the second selection shall determine the validity. The TOE must not accept the certificate if it fails any of the other validation rules in FIA_X509_EXT.1. If the administrator-configured option is selected by the ST Author, the ST Author must also make the appropriate selection in FMT_SMF.1.

Assurance Activity:

The evaluator shall check the TSS to ensure that it describes how the TOE/platform chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE/platform can use the certificates. If this functionality is implemented entirely by the platform, the operational guidance for the TOE shall reference the applicable guidance for each platform.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE/platform when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed. If this behavior is implemented entirely by the platform, the evaluator shall examine the ST of each platform to confirm that the selections for this element are contained in each platform's ST.

If this requirement is fully or partially implemented by the TOE, the evaluator shall perform Test 1 for each function in the system that requires the use of certificates:

*Test 1: The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in **FIA_X509_EXT.2.2** is performed. If the selected action is administrator-configurable, then the evaluator shall follow the operational guidance to determine that all supported administrator-configurable options behave in their documented manner.*

FIA_X509_EXT.2.3 The [selection: TOE, TOE platform] shall not establish an SA if a certificate or certificate path is deemed invalid.

Assurance Activity:

*Assurance activities for this element are tested through assurance activities for **FCS_IPSEC_EXT.1.12.***

4.2.4 Class: Security Management (FMT)

As indicated in Section 1 of this PP, the TOE is not required to maintain a separate management role. They are, however, required to provide functionality to configure certain aspects of TOE operation that should not be available to the general user population. If the TOE does provide some degree of administrative control, then the appropriate requirements from Appendix C should be used in the ST.

Specification of Management Functions (FMT_SMF)

FMT_SMF.1	**Specification of Management Functions**
FMT_SMF.1.1	The [selection: TOE, TOE platform, VPN Gateway] shall be capable of performing the following management functions: • **Configuration of IKE protocol version(s) used,** • **Configure IKE authentication techniques used,** • **Configure the cryptoperiod for the established session keys. The unit of measure for configuring the cryptoperiod shall be no greater than an hour,** • **Configure certificate revocation check,** • **Specify the algorithm suites that may be proposed and accepted during the IPsec exchanges,** • **load X.509v3 certificates used by the security functions in this PP,** • **ability to update the TOE, and to verify the updates,** • **ability to configure all security management functions identified in other sections of this PP,** • **[selection: action to be taken when a connection to determine the validity of a certificate cannot be established, [assignment: any additional management functions], no other actions].**

Application Note:

For installation, the VPN Client relies on the operational environment to authenticate the administrator to the client machine.

For the function "configure the cryptoperiod for the established session keys", the unit of measure for configuring the cryptoperiod shall be no greater than an hour. For example: units of measure in seconds, minutes and hours are acceptable and units of measure in days or greater are not acceptable.

There may be some instances where a VPN Gateway "pushes" configuration information down to the VPN Clients. This is an acceptable form of management, the ST Author simply must make clear in the ST what management functions are performed by the VPN Client, on the platform the VPN Client resides, and which are performed by the VPN Gateway. It may be the case that the functions overlap (i.e., can be done by an end-user on the platform or by the Gateway) and this is fine as long as the ST is clear and the guidance documentation describes how to perform the functions.

Assurance Activity:

The evaluator shall check to make sure that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function. The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE and testing each option listed in the requirement above. In cases where the management function is provided entirely by the platform (meaning that it is not able to be invoked by or through the TOE), the evaluator may simply ensure that the function is included in each underlying platform's ST.

As stated in the application note, a TOE may be configured either locally (through functions included in the VPN client itself or on its platform), or remotely by a VPN Gateway. The ST will clearly state which functions can be performed locally and remotely. The operational guidance documentation will describe how this is performed as well. The evaluator is expected to test this functions in all the ways in which the ST and guidance documentation state the configuration can be managed (with the exception noted in the previous paragraph).

Note that the testing here may be accomplished in conjunction with the testing of other requirements, such as FCS_IPSEC_EXT.1.

4.2.5 Class: Protection of the TSF (FPT)

Extended: TSF Self Test (FPT_TST_EXT)

FPT_TST_EXT.1 Extended: TSF Self Test

FPT_TST_EXT.1.1 The [selection: TOE, 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 The [selection: TOE, 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 [assignment: cryptographic services implemented as per the requirements in FCS_COP].

Application Note:

While the TOE is typically a software package running in the IT Environment, it is still capable of performing the self-test activities required above. It should be understood, however, that there is a significant dependency on the host environment in assessing the assurance provided by the tests mentioned above (meaning that if the host environment is compromised, the self tests will not be meaningful).

Cryptographic verification of the integrity is required, but the method by which this can be accomplished is specified in the ST in the assignment. The ST author will fill in the assignment with references to the cryptographic functions used to perform the integrity checks; this will include hashing and may potentially include digital signatures signed using X.509 certificates. All relevant FCS_COP requirements will be identified in the assignment by the ST author.

Assurance Activity:

Except for where it is explicitly noted, the evaluator is expected to check the following information regardless of whether the functionality is implemented by the TOE or by the TOE platform.

The evaluator shall examine the TSS to ensure that it details the self tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly. If some of the tests are performed by the TOE platform, the evaluator shall check the TSS to ensure that those tests are identified, and that the ST for each platform contains a description of those tests. Note that the tests that are required by this component are those that support security functionality in this PP, which may not correspond to the set of all self-tests contained in the platform STs.

The evaluator shall examine the TSS to ensure that it describes how the integrity of stored TSF executable code is cryptographically verified when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator shall check to ensure that the cryptographic requirements listed are consistent with the description of the integrity verification process.

The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases. For checks implemented entirely by the platform, the evaluator ensures that the operational guidance for the TOE references or includes the platform-specific guidance for each platform listed in the ST. The evaluator shall perform the following tests:

- *Test 1: The evaluator performs the integrity check on a known good TSF executable and verifies that the check is successful.*
- *Test 2: The evaluator modifies the TSF executable, performs the integrity check on the modified TSF executable and verifies that the check fails.*

Extended: Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1	**Extended: Trusted Update**
FPT_TUD_EXT.1.1	The [selection: TOE, TOE platform] shall provide the ability to query the current version of the TOE firmware/software.
FPT_TUD_EXT.1.2	The [selection: TOE, TOE platform] shall provide the ability to initiate updates to TOE firmware/software.
FPT_TUD_EXT.1.3	The [selection: TOE, TOE platform] shall provide a means to verify firmware/software updates to the TOE using a digital signature mechanism and [selection: published hash, no other functions] prior to installing those updates.

Application Note:
The digital signature mechanism referenced in the third element is the one specified in FCS_COP.1(2). The published hash referenced is generated by one of the functions specified in FCS_COP.1(3).

Assurance Activity:

Updates to the TOE are signed by an authorized source and may also have a hash associated with them, or are signed by an authorized source. If digital signatures are used, the definition of an authorized source is contained in the TSS, along with a description of how the certificates used by the update verification mechanism are contained on the device. The evaluator ensures this information is contained in the TSS. The evaluator also ensures that the TSS (or the operational guidance) describes how the candidate updates are obtained; the processing associated with verifying the digital signature or calculating the hash of the updates; and the actions that take place for successful (hash or signature was verified) and unsuccessful (hash or signature could not be verified) cases. If these activities are performed entirely by the underlying platform, a reference to the ST of each platform indicating that the required functionality is included for each platform shall be verified by the evalautor.

The evaluator shall perform the following tests (regardless of whether the functionality is implemented by the TOE or by the platform):

- *Test 1: The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains a legitimate update using procedures described in the operational guidance and verifies that it is successfully installed on the TOE. Then, the evaluator performs a subset of other assurance activity tests to demonstrate that the update functions as expected. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update.*

- *Test 2: The evaluator performs the version verification activity to determine the current version of the product. The evaluator obtains or produces an illegitimate update, and attempts to install it on the TOE. The evaluator verifies that the TOE rejects the update.*

4.2.6 Class: Trusted Path/Channels (FTP)

Trusted Channel (FTP_ITC)

FTP_ITC.1	**Inter-TSF trusted channel**
FTP_ITC.1.1	**Refinement:** The [selection: TOE, TOE platform] shall **use IPsec** to provide a **trusted** communication channel between itself and **a VPN Gateway** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data **from disclosure and detection of modification of the channel data**.
FTP_ITC.1.2	The [selection: TOE, TOE platform] shall permit *the TSF* to initiate communication via the trusted channel.
FTP_ ITC.1.3	The [selection: TOE, TOE platform] shall initiate communication via the trusted channel *for all traffic traversing that connection*.

Application Note:
The intent of the above requirement is to use the cryptographic protocols identified in the requirement to protect communications between the TOE and a VPN Gateway, both of which act as peers in the protocol sense.

The requirement implies that not only are communications protected when they are initially established, but also on resumption after an outage. It may be the case that some part of the TOE setup involves manually setting up tunnels to protect other communication, and if after an outage the TOE attempts to re-establish the communication automatically with (the necessary) manual intervention, there may be a window created where an attacker might be able to gain critical information or compromise a connection.

Assurance Activity:
The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to a VPN Gateway in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specification. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST. The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to the access point, and that it contains recovery instructions should a connection be unintentionally broken. The evaluator shall also perform the following tests:

- *Test 1: The evaluators shall ensure that the TOE is able to initiate communications with a VPN Gateway using the protocols specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- *Test 2: The evaluator shall ensure, for each communication channel with a VPN Gateway, the channel data is not sent in plaintext.*
- *Test 3: The evaluator shall ensure, for each communication channel with a VPN Gateway, modification of the channel data is detected by the TOE.*
- *Test 4: The evaluators shall physically interrupt the connection from the TOE to the a VPN Gateway. The evaluators shall ensure that subsequent communications are appropriately protected, at a minimum in the case of any attempts to automatically resume the connection or connect to a new access point.*

Further assurance activities are associated with the specific protocols.

4.3 Security Assurance Requirements

The Security Objectives for the TOE in Section 3 were constructed to address threats identified in Section 2. The Security Functional Requirements (SFRs) in Sections 4.1 and 4.2 are a formal instantiation of the Security Objectives.

While this section contains the complete set of SARs from the CC, the Assurance Activities to be performed by an evaluator are detailed in Sections 4.1 and 4.2 as well as in this section.

For each family, “Developer Notes” are provided on the developer action elements to clarify what, if any, additional documentation/activity needs to be provided by the developer. For the content/presentation and evaluator activity elements, additional assurance activities (to those already contained in Sec Sections 4.1 and 4.2) are described as a whole for the family, rather than for each element. Additionally, the assurance activities described in this section are complementary to those specified in Sections 4.1 and 4.2.

The TOE security assurance requirements, summarized in Table 10, identify the management and evaluative activities required to address the threats identified in Section 2 of this PP. Section 4.4 provides a succinct justification for choosing this set of assurance requirements for this PP.

Table 1: TOE Security Assurance Requirements

Assurance Class	Assurance Components	Assurance Components Description
Development	ADV_FSP.1	Basic Functional Specification
Guidance Documents	AGD_OPE.1	Operational user guidance
	AGD_PRE.1	Preparative User guidance
Tests	ATE_IND.1	Independent testing - conformance
Vulnerability Assessment	AVA_VAN.1	Vulnerability analysis
Life Cycle Support	ALC_CMC.1	Labeling of the TOE
	ALC_CMS.1	TOE CM coverage

4.3.1 Class ADV: Development

For TOEs conforming to this PP, the information about the TOE is contained in the guidance documentation available to the end user as well as the TOE Summary Specification (TSS) portion of the ST. While it is not required that the TOE developer write the TSS, the TOE developer must concur with the description of the product that is contained in the TSS as it relates to the functional requirements. The Assurance Activities contained in Section 4.1 should provide the ST authors with sufficient information to determine the appropriate content for the TSS section.

4.3.1.1 ADV_FSP.1 Basic functional specification

The functional specification describes the TOE Security Function Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces. Additionally, because TOEs conforming to this PP will necessarily have interfaces to the operational environment that are not directly invokable by TOE users (to include administrative users), there is little point specifying that such interfaces be described in and of themselves since only indirect testing of such interfaces may be possible. The activities for this family for this PP should focus on understanding the interfaces presented in the TSS in response to the functional requirement, and the interfaces presented in the AGD documentation. No additional "functional specification" document should be necessary to satisfy the assurance activities specified.

In understanding the interfaces to the TOE, it is important to consider that the threat that is to be countered is the confidentiality and integrity of the user data transmitted across the network (either via a TOE peer-to-peer connection, or TOE to VPN Gateway connection), as well as any authentication data that may traverse the connection. Additionally, the TOE, depending on its configuration, may offer protection of unauthorized access to the network behind the TOE. In addition to the network interface, the administrative interface (how the TOE is configured) also needs to be described.

The interfaces that need to be evaluated are characterized through the information needed to perform the assurance activities listed, rather than as an independent, abstract list.

Developer action elements:

ADV_FSP.1.1D	The developer shall provide a functional specification.
ADV_FSP.1.2D	The developer shall provide a tracing from the functional specification to the SFRs.
Developer Note:	As indicated in the introduction to this section, the functional specification is comprised of the information contained in the AGD_OPR and AGD_PRE documentation, coupled with the information provided in the TSS of the ST. The assurance activities in the functional requirements point to evidence that should exist in the documentation and TSS section; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

Content and presentation elements:

ADV_FSP.1.1C	The functional specification shall describe the purpose and method of use for each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.2C	The functional specification shall identify all parameters associated with each SFR-enforcing and SFR-supporting TSFI.
ADV_FSP.1.3C	The functional specification shall provide rationale for the implicit categorization of interfaces as SFR-non-interfering.
ADV_FSP.1.4C	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

Evaluator action elements:

ADV_ FSP.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
ADV_ FSP.1.2E	The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs.

Assurance Activity:

There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Sections 4.1 and 4.2, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the

functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

4.3.2 Class AGD: Guidance Documents

The guidance documents will be provided with the developer's security target. Guidance must include a description of the administrative model, and how the administrator verifies that the operational environment (the system that hosts the VPN Client) can fulfill its role for the security functionality. The documentation should be in an informal style and readable by an administrator.

Guidance must be provided for every operational environment that the product supports as claimed in the ST. This guidance includes:

- instructions to successfully install the TOE in that environment; and
- instructions to manage the security of the TOE as a product and as a component of the larger operational environment; and
- instructions to provide a protected administrative capability through the use of either TOE capabilities, environmental capabilities, or a combination of the two.

Guidance pertaining to particular security functionality must also be provided; requirements on such guidance are contained in the assurance activities specified in Sections 4.1 and 4.2.

4.3.2.1 AGD_OPE.1 Operational User Guidance

Developer action elements:

AGD_OPE.1.1D The developer shall provide operational user guidance.

Developer Note: Rather than repeat information here, the developer should review the assurance activities for this component to ascertain the specifics of the guidance that the evaluators will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

Content and presentation elements:

AGD_OPE.1.1C The operational user guidance shall describe, for each user role, the user-accessible functions and privileges that should be controlled in a secure processing environment, including appropriate warnings.

AGD_OPE.1.2C The operational user guidance shall describe, for each user role, how to use the available interfaces provided by the TOE in a secure manner.

AGD_OPE.1.3C The operational user guidance shall describe, for each user role, the available functions and interfaces, in particular all security parameters under the control of the user, indicating secure values as appropriate.

AGD_OPE.1.4C The operational user guidance shall, for each user role, clearly present each type of security-relevant event relative to the user-accessible functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_OPE.1.5C The operational user guidance shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AGD_OPE.1.6C The operational user guidance shall, for each user role, describe the security measures to be followed in order to fulfill the security objectives for the operational environment as described in the ST.

AGD_OPE.1.7C The operational user guidance shall be clear and reasonable.

Evaluator action elements:

AGD_OPE.1.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:
With respect to the management functions, while several have also been described in Sections 4.1 and 4.2, additional information is required as follows.

For TOE that implement a cryptographic engine, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The documentation must describe the process for verifying updates to the TOE, either by checking the hash or by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

- *For hashes, a description of where the hash for a given update can be obtained. For digital signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1(2) mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.*

- *Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).*

- *Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.*

4.3.2.2 AGD_PRE.1 Preparative Procedures

Developer action elements:

AGD_PRE.1.1D The developer shall provide the TOE including its preparative procedures.

Developer Note: As with the operational guidance, the developer should look to the assurance activities to determine the required content with respect to preparative procedures.

Content and presentation elements:

AGD_ PRE.1.1C The preparative procedures shall describe all the steps necessary for secure acceptance of the delivered TOE in accordance with the developer's delivery procedures.

AGD_ PRE.1.2C The preparative procedures shall describe all the steps necessary for secure installation of the TOE and for the secure preparation of the operational environment in accordance with the security objectives for the operational environment as described in the ST.

Evaluator action elements:

AGD_ PRE.1.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

AGD_ PRE.1.2E The evaluator *shall apply* the preparative procedures to confirm that the TOE can be prepared securely for operation.

Assurance Activity:
As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms and components (that is, combination of hardware and operating system) claimed for the TOE in the ST.

The evaluator shall check to ensure that the following guidance is provided:

- *As indicated in the introductory material, administration of the TOE is performed by one or more administrators that are a subset of the group of all users of the TOE. While it must be the case that the overall system (TOE plus Operational Environment) provide this capability, the responsibility for the implementation of the functionality can vary from totally the Operational Environment's responsibility to totally the TOE's responsibility. At a high level, the guidance must contain the appropriate instructions so that the Operational Environment is configured so that it provides the portion of the capability for which it is responsible. If the TOE provides no mechanism to allow separation of administrative users from the population of users, then the instructions, for instance, would cover the OS configuration of the OS I&A mechanisms to provide a unique (OS-based) identity for users, and further guidance would instruct the installer on the configuration of the DAC mechanisms of the OS using the TOE administrative identity (or identities) so that only TOE administrators would have access to the administrative executables. If the TOE provides some or all of this functionality, then the appropriate requirements are*

included in the ST from Appendix C, and the assurance activities associated with those requirements provide details on the guidance necessary for both the TOE and Operational Environment.

The evaluators shall also perform the following tests:

- *Test 1 [Conditional]: If the separation of administrative users from all TOE users is performed exclusively through the configuration of the Operational Environment, the evaluators will, for each configuration claimed in the ST, ensure that after configuring the system according to the administrative guidance, non-administrative users are unable to access TOE administrative functions.*

4.3.3 Class ATE: Tests

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this PP, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

4.3.3.1 ATE_IND.1 Independent Testing - Conformance

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operation) documentation provided. The focus of the testing is to confirm that the requirements specified in Sections 4.1 and 4.2 are being met, although some additional testing is specified for SARs in Section 4.3. The Assurance Activities identify the minimum testing activities associated with these components. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this PP.

	Developer action elements:
ATE_IND.1.1D	The developer shall provide the TOE for testing.
	Content and presentation elements:
ATE_IND.1.1C	The TOE shall be suitable for testing.
	Evaluator action elements:
ATE_IND.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.
ATE_IND.1.2E	The evaluator *shall test* a subset of the TSF to confirm that the TSF operates as specified.

Assurance Activity:

The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the body of this PP's Assurance Activities. While it

is not necessary to have one test case per test listed in an Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

The Test Plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platform and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluators are expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) is provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results. The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

4.3.4 Class AVA: Vulnerability Assessment

For the first generation of this protection profile, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, evaluators will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor. This information will be used in the development of penetration testing tools and for the development of future protection profiles.

4.3.4.1 AVA_VAN.1 Vulnerability Survey

	Developer action elements:
AVA_VAN.1.1D	The developer shall provide the TOE for testing.
	Content and presentation elements:
AVA_VAN.1.1C	The TOE shall be suitable for testing.
	Evaluator action elements:
AVA_VAN.1.1E	The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

AVA_VAN.1.2E The evaluator *shall perform* a search of public domain sources to identify potential vulnerabilities in the TOE.

AVA_VAN.1.3E The evaluator *shall conduct* penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

Assurance Activity:
As with ATE_IND, the evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to determine the vulnerabilities that have been found in VPN Client products in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. For example, if the vulnerability can be detected by pressing a key combination on boot-up, for example, a test would be suitable at the assurance level of this PP. If exploiting the vulnerability requires an electron microscope and a tank of liquid nitrogen, for instance, then a test would not be suitable and an appropriate justification would be formulated.

4.3.5 Class ALC: Life-cycle Support

At the assurance level provided for TOEs conformant to this PP, life-cycle support is limited to end-user-visible aspects of the life-cycle, rather than an examination of the TOE vendor's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation at this assurance level.

4.3.5.1 ALC_CMC.1 Labeling of the TOE

This component is targeted at identifying the TOE such that it can be distinguished from other products or version from the same vendor and can be easily specified when being procured by an end user.

Developer action elements:

ALC_CMC.1.1D The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.1.1C The TOE shall be labeled with its unique reference.

Evaluator action elements:

ALC_CMC.2.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:
The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

4.3.5.2 ALC_CMS.1 TOE CM coverage

Given the scope of the TOE and its associated evaluation evidence requirements, this component's assurance activities are covered by the assurance activities listed for ALC_CMC.1.

Developer action elements:

ALC_CMS.2.1D The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.2.1C The configuration list shall include the following: the TOE itself; and the evaluation evidence required by the SARs.

ALC_CMS.2.2C The configuration list shall uniquely identify the configuration items.

Evaluator action elements:

ALC_CMS.2.1E The evaluator *shall confirm* that the information provided meets all requirements for content and presentation of evidence.

Assurance Activity:
The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component.

4.4 Rationale for Security Assurance Requirements

The rationale for choosing these security assurance requirements is that they represent a more objective and repeatable level of assurance than has previously been accomplished using the generic CEM process. These assurance activities are, however, based on activities in the CEM, but tailored to this technology and specifically to the functional requirement contained in this PP. The assurance requirements and activities represent sufficient documentation and testing to mitigate the threats and achieve the objectives presented in the PP. If vulnerabilities are found in these types of products, then more stringent security assurance requirements may be mandated based on actual vendor practices.

Appendix A: References and Supporting Tables

In this Protection Profile, the focus in the initial sections of the document is to use a narrative presentation in an attempt to increase the overall understandability of the threats to network devices; the methods used to mitigate those threats; and the extent of the mitigation achieved by compliant TOEs. This presentation style does not readily lend itself to a formalized evaluation activity, so this Appendix—in addition to references --contains the tabular artifacts that can be used for the evaluation activities associated with this document.

A.1 References

[1] Common Criteria for Information Technology Security Evaluation (CC) Version 3.1, R3 July 2009

[2] Draft Consistency Instruction Manual, for Basic Robustness Environments, Release 4.0, CC version 3.1, 2008

[3] Federal Information Processing Standard Publication (FIPS-PUB) 140-2, Security Requirements for Cryptographic Modules, National Institute of Standards and Technology, May 25, 2001 (CHANGE NOTICES (12-03-2002)

[4] Federal Information Processing Standard Publication (FIPS-PUB) 180-3, Secure Hash Standard, October 2008

[5] Federal Information Processing Standard Publication (FIPS-PUB) 186-4, Digital Signature Standard (DSS), June 2009

[6] Federal Information Processing Standard Publication (FIPS-PUB) 197, Specification for the Advanced Encryption Standard (AES), November 26, 2001

[7] NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality, May 2004

[8] NIST Special Publication 800-57, Recommendation for Key Management, March 2007

[9] NIST Special Publication 800-63, Electronic Authentication Guideline, April 2006

[10] NIST Special Publication 800-90, Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) , March 2007

[11] NSA Glossary of Terms Used in Security and Intrusion Detection, Greg Stocksdale, NSA Information Systems Security Organization, April 1998. Need to update to CNSS 4009

[12] RFC 2865 Remote Authentication Dial In User Service (RADIUS), June 2000

[13] RFC 2868 RADIUS Attributes for Tunnel Protocol Support, June 2000

[14] RFC 3575 IANA Considerations for RADIUS, July 2003

[15] RFC 3579 RADIUS (Remote Authentication Dial In User Service Support For Extensible Authentication Protocol (EAP), September 2003

[16] RFC 3580 IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage

Guidelines, September 2003

[17] RFC 5216 The EAP-TLS Authentication Protocol, March 2008

[18] WPA2 Standard

A.2 Security Problem Description Tables

This section contains the tables supporting the descriptive information in Section 2. The rationale for the correspondence between the threats, objectives, and requirements is contained in the descriptive information in Sections 2 and 3.

A.2.1 Threats

The following table lists the threats addressed by the VPN Client and the operational environment. The assumed level of expertise of the attacker for all the threats identified below is unsophisticated.

Threat	Description of Threat
T.TSF_CONFIGURATION	Failure to allow configuration of the TSF may prevent its users from being able to adequately implement their particular security policy, leading to a compromise of user information.
T.TSF_FAILURE	Security mechanisms of the TOE may fail, leading to a compromise of the TSF.
T.UNAUTHORIZED_ACCESS	A user may gain unauthorized access to the TOE data. A malicious user, process, or external IT entity may masquerade as an authorized entity in order to gain unauthorized access to data or TOE resources. A malicious user, process, or external IT entity may misrepresent itself as the TOE to obtain identification and authentication data.
T.UNAUTHORIZED_UPDATE	A malicious party attempts to supply the end user with an update to the product that may compromise the security features of the TOE.
T.USER_DATA_REUSE	User data may be inadvertently sent to a destination not intended by the original sender because it is not rendered inaccessible after it is done being used.

A.2.2 Assumptions

These assumptions are made on the operational environment in order to be able to ensure that the security functionality specified in the PP can be provided by the TOE. If the TOE is placed in an operational environment that does not meet these assumptions, the TOE may no longer be able to provide all of its security functionality.

Assumption	Description of Assumption
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 operational environment will follow the applicable security configuration guidance.

A.3 Security Objectives Tables

A.3.1 Security Objectives for the TOE

Security Objectives for the TOE reflect the stated intent to counter identified threats. The TOE meets these objectives by satisfying the security functional requirements specified in the PP. As indicated in Section 1.1.3.1, some requirements in the PP may be satisfied by the Operational Environment rather than by the TOE, so technically objectives that are implemented or partially implemented in the Operational Environment would belong in the next section. It is expected that the ST Author will make the appropriate adjustments and clarifications when writing the ST. The end result should make it clear to the reader that all of the objectives have been satisfied, and it is clear where the requirements supporting those objectives are implemented (that is, by the VPN Client TOE or the platform on which it runs).

Objective	Objective Description
O.VPN_TUNNEL	The TOE will provide a network communication channel protected by encryption that ensures that the VPN client communicates with an authenticated VPN gateway.
O.RESIDUAL_INFORMATION_CLEARING	The TOE will ensure that any data contained in a protected resource is not available when the resource is reallocated.
O.TOE_ADMINISTRATION	The TOE will provide mechanisms to allow administrators to be able to configure the TOE.
O.TSF_SELF_TEST	The TOE will provide the capability to test some subset of its security functionality to ensure it is operating properly.
O.VERIFIABLE_UPDATES	The TOE will provide the capability to help ensure that any updates to the TOE can be verified by the administrator to be unaltered and (optionally) from a trusted source.

A.3.2 Security Objectives for the Operational Environment

The Operational Environment of the TOE implements technical and procedural measures to assist the TOE in correctly providing its security functionality (which is defined by the security objectives for the

TOE). This section defines the security objectives that are to be addressed by the IT domain or by non-technical or procedural means. As indicated above, if requirements supporting an objective on the TOE (in the previous table) are implemented in whole or in part by the platform, the ST should indicate this by an entry in this table with that objective.

Objective	Objective Description
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 operational environment.
OE.TRUSTED_CONFIG	Personnel configuring the TOE and its operational environment will follow the applicable security configuration guidance.

Appendix B: Optional Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. Additionally, there are three other types of requirements specified in Appendices B, C, and D.

The first type (in this Appendix) are requirements that can be included in the ST, but do not have to be in order for a TOE to claim conformance to this PP. The second type (in Appendix C) are requirements based on selections in the body of the PP: if certain selections are made, then additional requirements in that appendix will need to be included. The third type (in Appendix D) are components that are not required in order to conform to this PP, but will be included in the baseline requirements in future versions of this PP, so adoption by VPN Client vendors is encouraged. Note that the ST author is responsible for ensuring that requirements that may be associated with those in Appendix B, Appendix C, and/or Appendix D but are not listed (e.g., FMT-type requirements) are also included in the ST.

No optional items have been identified at this time.

Appendix C: Selection-Based Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements based on selections in the body of the PP: if certain selections are made, then additional requirements below will need to be included.

C.1 FIA_PSK_EXT (Extended: Pre-Shared Key Composition)

The TOE may support pre-shared keys for use in the IPsec protocol, and may use pre-shared keys in other protocols as well. There are two types of pre-shared keys that must be supported by the TOE, as specified in the requirements below. The first type is referred to as “text-based pre-shared keys”, which refer to pre-shared keys that are entered by users as a string of characters from a standard character set, similar to a password. Such pre-shared keys must be conditioned so that the string of characters is transformed into a string of bits, which is then used as the key.

The second type is referred to as “bit-based pre-shared keys” (for lack of a standard term); this refers to keys that are either generated by the TSF on a command from the administrator, or input in "direct form" by an administrator. "Direct form" means that the input is used directly as the key, with no "conditioning" as was the case for text-based pre-shared keys. An example would be a string of hex digits that represent the bits that comprise the key.

The requirements below mandate that the TOE must support both text-based and bit-based pre-shared keys, although generation of the bit-based pre-shared keys may be done either by the TOE or in the operational environment.

FIA_PSK_EXT.1	**Extended: Pre-Shared Key Composition**
FIA_PSK_EXT.1.1	The [selection: TOE, TOE platform] shall be able to use pre-shared keys for IPsec.
FIA_PSK_EXT.1.2	The [selection: TOE, TOE platform] shall be able to accept text-based pre-shared keys that: • are 22 characters and *[selection: [assignment: other supported lengths], no other lengths]*; • composed of any combination of upper and lower case letters, numbers, and special characters (that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”).
FIA_PSK_EXT.1.3	The [selection: TOE, TOE platform] shall [selection: condition the text-based pre-shared keys by using [selection: SHA-1, SHA-256, SHA-512, [assignment: *method of conditioning text string*]]; be able to [selection: accept, generate using the random bit generator specified in FCS_RBG_EXT.1] bit-based pre-shared keys].

Application Note:

For the length of the text-based pre-shared keys, a common length (22 characters) is required to help promote interoperability. If other lengths are supported they should be listed in the assignment; this assignment can also specify a range of values (e.g., "lengths from 5 to 55 characters") as well.

In the second selection for FIA_PSK_EXT.1.3, the ST author specifies the types of pre-shared keys that are supported. If "text-based pre-shared keys" is selected, the ST author fills in the method by which the text string entered by the administrator is "conditioned" into the bit string used as the key. This can be done by using one of the specified hash functions, or some other method through the assignment statement. If "bit-based pre-shared keys" is selected, the ST author specifies whether the TSF merely accepts bit-based pre-shared keys, or is capable of generating them. If it generates them, the requirement specified that they must be generated using the RBG specified by the requirements.

Assurance Activity:

Requirement met by the platform

For each platform listed in the ST, the evaluator shall examine the ST of the platform to ensure that the functions associated with pre-shared keys claimed in that platform's ST contains the same functions specified in the VPN Client's ST. If the TOE does not perform any management or input of the pre-shared keys then no further activity is required; however, any management functions related to pre-shared keys that is performed by the TOE must be specified in the TOE's operational guidance and verified by the evaluator.
Regardless of whether this capability is implemented by the TOE or by the platform, the tests listed in the "Requirement met by the TOE" section must still be performed for each platform claimed in the ST.

Requirement met by the TOE

The evaluator shall examine the operational guidance to determine that it provides guidance on the composition of strong text-based pre-shared keys, and (if the selection indicates keys of various lengths can be entered) that it provides information on the merits of shorter or longer pre-shared keys. The guidance must specify the allowable characters for pre-shared keys, and that list must be a super-set of the list contained in FIA_PSK_EXT.1.2.

The evaluator shall examine the TSS to ensure that it states that text-based pre-shared keys of 22 characters are supported. If "text-based pre-shared keys" is selected, the evaluator shall confirm that the TSS states the conditioning that takes place to transform the text-based pre-shared key from the key sequence entered by the user (e.g., ASCII representation) to the bit string used by IPsec, and that this conditioning is consistent with the last selection in the FIA_PSK_EXT.1.3 requirement.

If "bit-based pre-shared keys" is selected, the evaluator shall confirm the operational guidance contains instructions for either entering bit-based pre-shared keys for each protocol identified in the requirement, or generating a bit-based pre-shared key (or both). The evaluator shall also examine the TSS to ensure it describes the process by which the bit-based pre-shared keys are generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1.

The evaluator shall also perform the following tests:

- *Test 1: The evaluator shall compose a pre-shared key of 22 characters that contains a combination of the allowed characters in accordance with the operational guidance, and demonstrates that a successful protocol negotiation can be performed with the key.*

- *Test 2 [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator shall repeat Test 1 using the minimum length; the maximum length; and an invalid length. The minimum and maximum length tests should be successful, and the invalid length must be rejected by the TOE.*

- *Test 3 [conditional]: If the TOE does not generate bit-based pre-shared keys, the evaluator shall obtain a bit-based pre-shared key of the appropriate length and enter it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.*

- *Test 4 [conditional]: If the TOE does generate bit-based pre-shared keys, the evaluator shall generate a bit-based pre-shared key of the appropriate length and use it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.*

Appendix D: Objective Requirements

As indicated in the introduction to this PP, the baseline requirements (those that must be performed by the TOE or its underlying platform) are contained in the body of this PP. There are additional requirements that specify security functionality that is desirable and these requirements are contained in this Annex. It is expected that these requirements will transition from objective requirements to baseline requirements in future versions of this PP.

D.1 FAU (Security Audit)

Security audit data generation (FAU_GEN)

FAU_GEN.1	**Audit Data Generation**
FAU_GEN.1.1	The TSF shall be able to generate an audit record of the following auditable events: a) Start-up and shutdown of the audit functions; b) All auditable events for the not specified level of audit; and c) *All* administrative actions; d) [*Specifically defined auditable events listed in Table3*].

Application Note:
The ST author can include other auditable events directly in the table; they are not limited to the list presented.

In the case of "a", the audit functions referred to are those provided by the TOE. For example, in the case that the TOE was a stand-alone executable, auditing the startup and the shutdown of the TOE itself would be sufficient to meet the requirements of this clause.

Many auditable aspects of the SFRs included in this document deal with administrative actions. Item c above requires all administrative actions to be auditable, so no additional specification of the auditability of these actions is present in Table 9. While the TOE itself does not need to provide the ability to perform I&A for an administrator, this requirement implies that the TOE possess the capability to audit the events described by the PP as "administrative actions" (primarily dealing with configuration of the functionality provided by the TOE). It is expected that the AGD guidance detail the steps needed to ensure the audit data generated by the TOE is integrated with the audit capabilities of the underlying IT environment.

Assurance Activity:
The evaluator shall check the operational guidance and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the PP is described and that the description of the fields contains the information required in FAU_GEN.1.2, and the additional information specified in Table 9.

The evaluator shall in particular ensure that the operational guidance is clear in relation to the contents for failed cryptographic events. In Table 9, information detailing the cryptographic mode of operation and a name or identifier for the object being encrypted is required. The evaluator shall ensure that name

or identifier is sufficient to allow an administrator reviewing the audit log to determine the context of the cryptographic operation (for example, performed during a key negotiation exchange, performed when encrypting data for transit) as well as the non-TOE endpoint of the connection for cryptographic failures relating to communications with other IT systems.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP. The TOE may contain functionality that is not evaluated in the context of this PP because the functionality is not specified in an SFR. This functionality may have administrative aspects that are described in the operational guidance. Since such administrative actions will not be performed in an evaluated configuration of the TOE, the evaluator shall examine the operational guidance and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP, which thus form the set of "all administrative actions". The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.

The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records in accordance with the assurance activities associated with the functional requirements in this PP. Additionally, the evaluator shall test that each administrative action applicable in the context of this PP is auditable. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.

FAU_GEN.1.2	The [TOE, TOE Platform] shall record within each audit record at least the following information: a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [*information specified in column three of Table 2 below]*.

Application Note:
As with the previous component, the ST author should update Table 2 with any additional information generated. "Subject identity" in the context of this requirement could either be the administrator's user id or the affected network interface, for example.

Assurance Activity:

This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

Table 2: Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	None.
FCS_CKM.1(*)	Failure of the key generation activity.	None.
FCS_CKM_EXT.2	None.	None.
FCS_CKM_EXT.4	Failure of the key zeroization process.	Identity of object or entity being cleared.
FCS_COP.1(1)	Failure of encryption or decryption.	Cryptographic mode of operation, name/identifier of object being encrypted/decrypted.
FCS_COP.1(2)	Failure of cryptographic signature.	Cryptographic mode of operation, name/identifier of object being signed/verified.
FCS_COP.1(3)	Failure of hashing function.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_COP.1(4)	Failure in Cryptographic Hashing for Non-Data Integrity.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_IPSEC_EXT.1	Decisions to DISCARD, BYPASS, PROTECT network packets processed by the TOE. Failure to establish an IPsec SA. Establishment/Termination of an IPsec SA.	Presumed identity of source subject. Identity of destination subject. Transport layer protocol, if applicable. Source subject service identifier, if applicable. The entry in the SPD that applied to the decision. Reason for failure. Non-TOE endpoint of connection (IP address) for both successes and failures.
FCS_RBG_EXT.1	Failure of the randomization process.	None.

Requirement	Auditable Events	Additional Audit Record Contents
FDP_IFC_EXT.1	Failure of the randomization process.	None.
FDP_RIP.2	None.	
FIA_PSK_EXT	Failure to establish exclusive tunnel.	None.
FIA_X509_EXT.1	Failure of the X.509 certificate validation.	Reason for failure of validation.
FIA_X509_EXT.1	None.	None.
FMT_SMF.1	Success or failure of function.	None.
FPT_TUD_EXT.1	Initiation of the update. Any failure to verify the integrity of the update.	No additional information.
FTP_ITC.1	All attempts to establish a trusted channel. Detection of modification of channel data.	Identification of the non-TOE endpoint of the channel.

Security Audit Event Selection (FAU_SEL)

FAU_SEL.1 **Selective Audit**

FAU_SEL.1.1 The TSF shall be able to select the set of events to be audited from the set of all auditable events based on the following attributes:

a) **event type;**
b) **success of auditable security events;**
c) **failure of auditable security events; and**
d) **[assignment: other attributes].**

Application Note:
The intent of this requirement is to identify all criteria that can be selected to trigger an audit event. This can be configured through an interface on the client for a user/administrator to invoke, or it could be an interface that the VPN Gateway uses to instruct the client on which events are to be audited. For the ST author, the assignment is used to list any additional criteria or "none". The auditable event types are listed in Table 9.

Assurance Activity:
The evaluator shall review the administrative guidance to ensure that the guidance itemizes all event types, as well as describes all attributes that are to be selectable in accordance with the requirement, to include those attributes listed in the assignment. The administrative guidance shall also contain instructions on how to set the pre-selection, or how the VPN Gateway will configure the client, as well as

explain the syntax (if present) for multi-value pre-selection. The administrative guidance shall also identify those audit records that are always recorded, regardless of the selection criteria currently being enforced.

The evaluator shall also perform the following tests:

- *Test 1: For each attribute listed in the requirement, the evaluator shall devise a test to show that selecting the attribute causes only audit events with that attribute (or those that are always recorded, as identified in the administrative guidance) to be recorded.*

Test 2 [conditional]: If the TSF supports specification of more complex audit pre-selection criteria (e.g., multiple attributes, logical expressions using attributes) then the evaluator shall devise tests showing that this capability is correctly implemented. The evaluator shall also, in the test plan, provide a short narrative justifying the set of tests as representative and sufficient to exercise the capability.

D.2 FDP_IFC (Subset Information Flow Control)

FDP_IFC_EXT.1 Subset Information Flow Control

FDP_IFC_EXT.1.1 The TSF shall ensure that **all IP traffic (other than IP traffic required to establish the VPN connection) to flow through the IPsec VPN client**.

Application Note:

This requirement is used when the VPN client is able to enforce the requirement through its own components. This generally will have to be done through using hooks provided by the platform such that the TOE is able to ensure that no IP traffic can flow through other network interfaces.

Assurance Activity:

The evaluator shall verify that the TSS section of the ST describes the routing of IP traffic through processes on the TSF when a VPN client is enabled. The evaluator shall ensure that the description indicates which traffic does not go through the VPN and which traffic does and that a configuration exists for each baseband protocol in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec). The ST author shall also identify in the TSS section any differences in the routing of IP traffic when using any supported baseband protocols (e.g. WiFi or, LTE).

The evaluator shall verify that the following is addressed by the documentation:

- *The description above indicates that if a VPN client is enabled, all configurations route all IP traffic (other than IP traffic required to establish the VPN connection) through the VPN client.*
- *The AGD guidance describes how the user and/or administrator can configure the TSF to meet this requirement.*

The evaluator shall also perform the following tests.

Test 1: If the ST author identifies any differences in the routing between WiFi and cellular protocols, the evaluator shall repeat this test with a base station implementing one of the identified cellular protocols.

Step 1 - The evaluator shall enable a WiFi configuration as described in the AGD guidance. The evaluator shall use a packet sniffing tool between the platform and an Internet-connected network. The evaluator shall turn on the sniffing tool and perform actions with the device such as navigating to websites, using provided applications, and accessing other Internet resources. The evaluator shall verify that the sniffing tool captures the traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 2 -The evaluator shall configure an IPsec VPN client that supports the routing specified in this requirement, and if necessary, configure the device to perform the routing specified as described in the AGD guidance. The evaluator shall turn on the sniffing tool, establish the VPN connection, and perform the same actions with the device as performed in the first step. The evaluator shall verify that the sniffing tool captures traffic generated by these actions, turn off the sniffing tool, and save the session data.

Step 3 - The evaluator shall examine the traffic from both step one and step two to verify that all IP traffic, aside from and after traffic necessary for establishing the VPN (such as IKE, DNS, and possibly HTTPS), is encapsulated by IPsec. The evaluator shall be aware that IP traffic on the cellular baseband outside of the IPsec tunnel may be emanating from the baseband processor and shall verify with the manufacturer that any identified traffic is not emanating from the application processor.

Step 4 - The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local wireless network, and shall verify that the TOE discards them.

Appendix E: Entropy Documentation and Assessment

The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing. This documentation is not required to be part of the TSS.

Design Description

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. It will describe the operation of the entropy source to include how it works, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the random comes from, where it is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate.

Entropy Justification

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source exhibiting probabilistic behavior (an explanation of the probability distribution and justification for that distribution given the particular source is one way to describe this). This argument will include a description of the expected entropy rate and explain how you ensure that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

Operating Conditions

Documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

Health Testing

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and

rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source.

Appendix F: Glossary and Acronyms

F.1 Glossary

Administrator – a user that has administrative privilege to configure the TOE in privileged mode.

Authentication Server (AS) – an entity designed to facilitate the authentication of an entity (user or client) that attempts to access a protected network.

Authorized – an entity granted access privileges to an object, system or system entity.

Critical Security Parameter (CSP) – security related information, e.g. secret and private cryptographic keys, and authentication data such as passwords and PINs, whose disclosure or modification can compromise the security of a cryptographic module.

Entropy Source – this cryptographic function provides a seed for a random number generator by accumulating the outputs from one or more noise sources. The functionality includes a measure of the minimum work required to guess a given output and tests to ensure that the noise sources are operating properly.

FIPS-approved cryptographic function – a security function (e.g., cryptographic algorithm, cryptographic key management technique, or authentication technique) that is either: 1) specified in a Federal Information Processing Standard (FIPS), or 2) adopted in a FIPS and specified either in an appendix to the FIPS or in a document referenced by the FIPS.

IT Environment – hardware and software that are outside the TOE boundary that support the TOE functionality and security policy.

Operational Environment – the environment in which the TOE is operated.

Private Network – a network that is protected from access by unauthorized users or entities.

Privileged Mode – a TOE operational mode that allows a user to perform functions that require IT Environment administrator privileges.

Public Network – a network that is visible to all users and entities and does not protect against unauthorized access (e.g. internet).

Security Assurance Requirement (SAR) – description of how assurance is to be gained that the TOE meets the SFR.

Security Functional Requirement (SFR) – translation of the security objectives for the TOE into a standardized language.

Security Target (ST) – implementation-dependent statement of security needs for a specific identified TOE.

Target of Evaluation (TOE) – set of software, firmware and/or hardware possibly accompanied by guidance. For this PP the TOE is the VPN Client.

Threat Agent - an entity that tries to harm an information system through destruction, disclosure, modification of data, and/or denial of service.

TOE Security Functionality (TSF) – combined functionality of all hardware, software, and firmware of a TOE that must be relied upon for the correct enforcement of the SFRs.

TOE Summary Specification (TSS) – a description of how the TOE satisfies all of the SFRs.

Unauthorized User – an entity (device or user) who has not been authorized by an authorized administrator to access the TOE or private network.

Unprivileged Mode – a TOE operational mode that only provides VPN client functions for the VPN Client user.

VPN Client – the TOE, allows remote users to use client computers to establish an encrypted IPsec tunnel across an unprotected public network to a private network

VPN Client User – a user operating the TOE in unprivileged mode.

VPN Gateway – a component that performs encryption and decryption of IP packets as they cross the boundary between a private network and a public network

F.2 Acronyms

AES	Advanced Encryption Standard
AF	Authorization factor
AS	Authorization subsystem
CAVS	Cryptographic Algorithm Validation System
CC	Common Criteria
CCTL	Common Criteria Testing Laboratory
CM	Configuration management
COTS	Commercial Off-The-Shelf
CMVP	Cryptographic Module Validation Program
DRBG	Deterministic Random Bit Generator
DoD	Department of Defense
EAL	Evaluation Assurance Level
ES	Encryption Subsystem
FIPS	Federal Information Processing Standards
ISSE	Information System Security Engineers
IT	Information Technology
OSP	Organization Security Policy
PP	Protection Profile
PUB	Publication

RBG	Random Bit Generator
SAR	Security Assurance Requirements
SF	Security Function
SFR	Security Functional Requirement
ST	Security Target
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFI	TSF Interface
TSS	TOE Summary Specification

Appendix G: PP Identification

Tile:	Protection Profile for IPsec Virtual Private Network (VPN) Clients
Version:	1.4
Sponsor:	National Information Assurance Partnership (NIAP)
CC Version:	Common Criteria for Information Technology Security Evaluation (CC) Version 3.1, R3 July 2009
Keywords:	Authentication Server , IKE, IPsec, PKI , VPN, VPN Client, VPN

Appendix H: Document Conventions

Except for replacing United Kingdom spelling with U.S. English spelling, the notation, formatting, and conventions used in this PP are consistent with version 3.1 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP reader.

The notation, formatting, and conventions used in this PP are largely consistent with those used in version 3.1 of the Common Criteria (CC). Selected presentation choices are discussed here to aid the PP user. The CC allows several operations to be performed on functional and assurance requirements; *refinement*, *selection*, *assignment*, and *iteration* are defined in Appendix C4 of Part 1 of the CC 3.1. Each of these operations is used in this PP.

Refinement Convention

The **refinement** operation is used to add detail to a requirement, and thus further restricts a requirement. Refinement of security requirements is denoted by the word "Refinement" in **bold text** after the element number and the additional text in the requirement in bold text.

Selection Convention

The **selection** operation is used to select one or more options provided by the CC in stating a requirement (see appendix C.4.3 Part 1, CC 3.1). Selections that have been made by the PP authors show the selection in **bold** characters, the brackets and the word "selection" removed. Selections to be filled in by the ST author are shown in square brackets with an indication that a selection is to be made, [selection:].

Assignment Convention

The **assignment** operation is used to assign a specific value to an unspecified parameter, such as the length of a password (see appendix C.4.2 Part 1, CC 3.1). Showing the value in **bold** characters denotes assignments that have been made by the PP authors, the brackets and the word "assignment" are removed. Assignments to be filled in by the ST author are shown in square brackets with an indication that an assignment is to be made [assignment:].

Iteration Convention

The **iteration** operation is used when a component is repeated with varying operations (see appendix C.4.1 Part 1, CC 3.1). The iteration number (iteration_number) is show in parenthesis following the component identifier.

The iteration operation may be performed on every component. The PP/ST author performs an iteration operation by including multiple requirements based on the same component. Each iteration of a component shall be different from all other iterations of that component, which is realized by completing assignments and selections in a different way, or by applying refinements to it in a different way.

Extended Requirement Convention

Extended requirements are permitted if the CC does not offer suitable requirements to meet the authors' needs. **Extended requirements** must be identified and are required to use the CC class/family/component model in articulating the requirements. Extended requirements will be indicated with the "EXT" inserted within the component.

Application Notes

Application notes contain additional supporting information that is considered relevant or useful for the construction of security targets for conformant TOEs, as well as general information for developers, evaluators, and ISSEs. Application notes also contain advice relating to the permitted operations of the component.

Assurance Activities

Assurance activities serve as a Common Evaluation Methodology for the functional requirements levied on the TOE to mitigate the threat. The activities include instructions for evaluators to analyze specific aspects of the TOE as documented in the TSS, thus levying implicit requirements on the ST author to include this information in the TSS section. In this version of the PP these activities are directly associated with the functional and assurance components, although future versions may move these requirements to a separate appendix or document.