Microsoft Windows

Common Criteria Evaluation

Microsoft Windows 10

Security Target

Document Information	
Version Number	0.07
Updated On	October 28, 2016

Version History

Version	Date	Summary of changes
0.01	April 29, 2016	Initial draft
0.02	May 13, 2016	Updates from evaluation
0.03	June 17, 2016	Updates from evaluation
0.04	June 27, 2016	Updates from evaluation
0.05	September 22, 2016	Updates from validation
0.06	October 14, 2016	Updates from validation
0.07	October 28, 2016	Updates for publication

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

Note: Green text is a required CSfC selection.

This section presents the following information required for a Common Criteria (CC) evaluation:

- Identifies the Security Target (ST) and the Target of Evaluation (TOE);
- Specifies the security target conventions and conformance claims; and,
- Describes the organization of the security target.

1.1 Security Target, TOE, and Common Criteria (CC) Identification

ST Title: Microsoft Windows 10 IPsec VPN Client Security Target

ST Version: version 0.07, October 28, 2016

TOE Software Identification: The following Windows Operating Systems (OS):

- Microsoft Windows 10 Pro Edition (64-bit version)
- Microsoft Windows 10 Enterprise Edition (64-bit version)

The following security updates and patches must be applied to the above Windows products:

- All critical updates as of May 30, 2016

TOE Platform Hardware Identification: The TOE, which is the Microsoft Windows 10 IPsec VPN client, is available on any general-purpose hardware which can run Windows 10, but the formal evaluation results are only applicable to x64 Intel-based platforms. The following computers were used for testing during the evaluation:

- Surface Pro 4
- Surface Book

TOE Guidance Identification: The following administrator, user, and configuration guides were evaluated as part of the TOE:

- *Microsoft Windows Common Criteria Supplemental Admin Guidance for IPsec VPN Clients* along with all the documents referenced therein.

Evaluation Assurance: As specified in section 5.2.1 and specific Assurance Activities associated with the security functional requirements from section 5.2.2.

CC Identification: CC for Information Technology (IT) Security Evaluation, Version 3.1, Revision 4, September 2012.

1.2 CC Conformance Claims

This TOE and ST are consistent with the following specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security functional requirements, Version 3.1, Revision 4, September 2012, extended (Part 2 extended)
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance requirements Version 3.1, Revision 4 September 2012, (Part 3 conformant)
- Protection Profile for IPsec Virtual Private Network (VPN) Clients, Version 1.4, October 21, 2013, (IPsec VPN Client PP)
- Evaluation Assurance Activities specified in section 5.2.2 and CC Part 3 assurance requirements specified in section 5.2.1

1.3 Conventions, Terminology, Acronyms

This section specifies the formatting information used in the security target.

1.3.1 Conventions

The following conventions have been applied in this document:

- Security Functional Requirements (SFRs): Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
    - Iteration: allows a component to be used more than once with varying operations.
    - Assignment: allows the specification of an identified parameter.
    - Selection: allows the specification of one or more elements from a list.
    - Refinement: allows the addition of details.

  The conventions for the assignment, selection, refinement, and iteration operations are described in Section 5.

- Other sections of the security target use a bold font to highlight text of special interest, such as captions.

1.3.2 Terminology

The following terminology is used in the security target:

Term	Definition
Access	Interaction between an entity and an object that results in the flow or modification of data.
Access control	Security service that controls the use of resources[1] and the disclosure and modification of data[2].
Accountability	Tracing each activity in an IT system to the entity responsible for the activity.

[1] Hardware and software

[2] Stored or communicated

Active Directory	Active Directory manages enterprise identities, credentials, information protection, system and application settings through AD Domain Services, Federation Services, Certificate Services and Lightweight Directory Services.
Administrator	An authorized user who has been specifically granted the authority to manage some portion or the entire TOE and thus whose actions may affect the TOE Security Policy (TSP). Administrators may possess special privileges that provide capabilities to override portions of the TSP.
Assurance	A measure of confidence that the security features of an IT system are sufficient to enforce the IT system's security policy.
Attack	An intentional act attempting to violate the security policy of an IT system.
Authentication	A security measure that verifies a claimed identity.
Authentication data	The information used to verify a claimed identity.
Authorization	Permission, granted by an entity authorized to do so, to perform functions and access data.
Authorized user	An authenticated user who may, in accordance with the TOE Security Policy, perform an operation.
Availability	Timely[3], reliable access to IT resources.
Compromise	Violation of a security policy.
Confidentiality	A security policy pertaining to disclosure of data.
Critical cryptographic security parameters	Security-related information appearing in plaintext or otherwise unprotected form and whose disclosure or modification can compromise the security of a cryptographic module or the security of the information protected by the module.
Cryptographic boundary	An explicitly defined contiguous perimeter that establishes the physical bounds (for hardware) or logical bounds (for software) of a cryptographic module.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines: • the transformation of plaintext data into ciphertext data • the transformation of ciphertext data into plaintext data • a digital signature computed from data • the verification of a digital signature computed from data • a data authentication code computed from data
Cryptographic module	The set of hardware, software, and/or firmware that implements approved security functions, including cryptographic algorithms and key generation, which is contained within the cryptographic boundary.
Cryptographic module security policy	A precise specification of the security rules under which a cryptographic module must operate.
Defense-in-depth	A security design strategy whereby layers of protection are utilized to establish an adequate security posture for an IT system.
Discretionary Access Control (DAC)	A means of restricting access to objects based on the identity of subjects and groups to which the objects belong. The controls are discretionary meaning that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject.

[3] According to a defined metric

Edition	A distinct variation of a Windows OS version. Examples of editions are Windows Server 2012 [Standard] and Windows Server 2012 Datacenter.
Enclave	A collection of entities under the control of a single authority and having a homogeneous security policy. They may be logical, or based on physical location and proximity.
Entity	A subject, object, user or external IT device.
General-Purpose Operating System	A general-purpose operating system is designed to meet a variety of goals, including protection between users and applications, fast response time for interactive applications, high throughput for server applications, and high overall resource utilization.
Identity	A means of uniquely identifying an authorized user of the TOE.
Integrated Windows authentication	An authentication protocol formerly known as NTLM or Windows NT Challenge/Response.
Named object	• An object that exhibits all of the following characteristics: • The object may be used to transfer information between subjects of differing user identities within the TOE Security Function (TSF). • Subjects in the TOE must be able to request a specific instance of the object. • The name used to refer to a specific instance of the object must exist in a context that potentially allows subjects with different user identities to request the same instance of the object.
Object	An entity under the control of the TOE that contains or receives information and upon which subjects perform operations.
Operating environment	The total environment in which a TOE operates. It includes the physical facility and any physical, procedural, administrative and personnel controls.
Persistent storage	All types of data storage media that maintain data across system boots (e.g., hard disk, removable media).
Public object	An object for which the TSF unconditionally permits all entities "read" access under the Discretionary Access Control SFP. Only the TSF or authorized administrators may create, delete, or modify the public objects.
Resource	A fundamental element in an IT system (e.g., processing time, disk space, and memory) that may be used to create the abstractions of subjects and objects.
SChannel	A security package (SSP) that provides network authentication between clients and servers.
Secure State	Condition in which all TOE security policies are enforced.
Security attributes	TSF data associated with subjects, objects and users that is used for the enforcement of the TSP.
Security-enforcing	A term used to indicate that the entity (e.g., module, interface, subsystem) is related to the enforcement of the TOE security policies.
Security-supporting	A term used to indicate that the entity (e.g., module, interface, subsystem) is not security-enforcing; however, the entity's implementation must still preserve the security of the TSF.
Security context	The security attributes or rules that are currently in effect. For SSPI, a security context is an opaque data structure that contains security data

	relevant to a connection, such as a session key or an indication of the duration of the session.
Security package	The software implementation of a security protocol. Security packages are contained in security support provider libraries or security support provider/authentication package libraries.
Security principal	An entity recognized by the security system. Principals can include human users as well as autonomous processes.
Security Support Provider (SSP)	A dynamic-link library that implements the SSPI by making one or more security packages available to applications. Each security package provides mappings between an application's SSPI function calls and an actual security model's functions. Security packages support security protocols such as Kerberos authentication and Integrated Windows Authentication.
Security Support Provider Interface (SSPI)	A common interface between transport-level applications. SSPI allows a transport application to call one of several security providers to obtain an authenticated connection. These calls do not require extensive knowledge of the security protocol's details.
Security Target (ST)	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Subject	An active entity within the TOE Scope of Control (TSC) that causes operations to be performed. Subjects can come in two forms: trusted and untrusted. Trusted subjects are exempt from part or all of the TOE security policies. Untrusted subjects are bound by all TOE security policies.
Target of Evaluation (TOE)	An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation.
Threat	Capabilities, intentions and attack methods of adversaries, or any circumstance or event, with the potential to violate the TOE security policy.
Unauthorized individual	A type of threat agent in which individuals who have not been granted access to the TOE attempt to gain access to information or functions provided by the TOE.
Unauthorized user	A type of threat agent in which individuals who are registered and have been explicitly granted access to the TOE may attempt to access information or functions that they are not permitted to access.
Universal Unique Identifier (UUID)	UUID is an identifier that is unique across both space and time, with respect to the space of all UUIDs. A UUID can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects across a network.
User	Any person who interacts with the TOE.
User Principal Name (UPN)	An identifier used by Microsoft Active Directory that provides a user name and the Internet domain with which that username is associated in an e-mail address format. The format is [*AD username*]@[associated *domain*]; an example would be *john.smith@microsoft.com.*
Uniform Resource Locator (URL)	The address that is used to locate a Web site. URLs are text strings that must conform to the guidelines in RFC 2396.
Version	A Version refers to a release level of the Windows operating system. Windows 7 and Windows 8 are different versions.
Vulnerability	A weakness that can be exploited to violate the TOE security policy.

1.3.3 Acronyms

The acronyms used in this security target are specified in **Appendix A: List of Abbreviations**.

1.4 ST Overview and Organization

The Windows TOE provides the following security services:

- Cryptographic support
- User data protection
- Identification and Authentication (I&A)
- Protection of the TOE Security Functions (TSF)
- Trusted path/channel
- Security management
- Audit

This security target contains the following additional sections:

- **TOE Description** (Section 2): Provides an overview of the TSF and boundary.
- **Security Problem Definition**Security Problem Definition (Section 3): Describes the threats, organizational security policies and assumptions that pertain to the TOE.
- **Security Objectives** (Section 4): Identifies the security objectives that are satisfied by the TOE and the TOE operational environment.
- **Security Requirements** (Section 5): Presents the security functional and assurance requirements met by the TOE.
- **TOE Summary Specification (TSS**) (Section 6): Describes the security functions provided by the TOE to satisfy the security requirements and objectives.
- **Protection Profile Conformance Claim** (Section 7): Presents the rationale concerning compliance of the ST with the ***Protection Profile for IPsec Virtual Private Network (VPN) Clients****.*
- **Rationale for Modifications to the Security Requirements** (Section 8): Presents the rationale for the security objectives, requirements, and TOE Summary Specification as to their consistency, completeness and suitability.

2 TOE Description

The TOE includes the Windows 10 operating system and those applications necessary to manage, support and configure the operating system in order to provide IPsec VPN Client capabilities to the user.

2.1 Product Types

Windows 10 is a preemptive multitasking, multiprocessor, and multi-user operating system. In general, operating systems provide users with a convenient interface to manage underlying hardware. They control the allocation and manage computing resources such as processors, memory, and Input/Output

(I/O) devices. Windows expands these basic operating system capabilities to controlling the allocation and managing higher level IT resources such as security principals (user or machine accounts), files, printing objects, services, window station, desktops, cryptographic keys, network ports traffic, directory objects, and web content. Multi-user operating systems such as Windows keep track of which user is using which resource, grant resource requests, account for resource usage, and mediate conflicting requests from different programs and users.

2.2 Product Description

The TOE includes two product variants of Windows 10:

- Windows 10 Pro
- Windows 10 Enterprise

Windows 10 is suited for business desktops, notebook, tablet, and convertible computers. It is the workstation product and while it can be used by itself, it is designed to serve as a client within Windows domains.

Windows 10 has undergone several Common Criteria evaluations including:

- Windows 10 Mobile Device Fundamentals Common Criteria evaluation
- Windows 10 General Purpose OS Common Criteria evaluation
- Microsoft Windows 10 Mobile with Lumia 950, 950 XL, 550, 635, and Windows 10 with Surface Pro 4
- Microsoft Windows 10 with Surface Book

2.3 Security Environment and TOE Boundary

The TOE includes both physical and logical boundaries. Its operational environment is that of a networked environment with IEEE 802.11 (Wi-Fi) or a local area network.

2.3.1 Logical Boundaries

The logical boundary of the TOE includes:

- The **Boot Manager**, which is invoked by the computer’s bootstrapping code.
- The **Windows Loader** which loads the operating system into the computer’s memory.
- **Windows OS Resume** which reloads an image of the executing operating system from a hibernation file as part of resuming from a hibernated state.
- The **Windows Kernel** which contains device drivers for the Windows NT File System, full volume encryption, the crash dump filter, and the kernel-mode cryptographic library.
- The **IPv4 / IPv6 network stack** in the kernel.
- The **IPsec** module in user-mode.

- The **IKE and AuthIP Keying Modules** service which hosts the IKE and Authenticated Internet Protocol (AuthIP) keying modules. These keying modules are used for authentication and key exchange in Internet Protocol security (IPsec).[4]
- The **Remote Access Service** device driver in the kernel, which is used primarily for ad hoc or user-defined VPN connections; known as the "RAS IPsec VPN" or "RAS VPN".
- The **IPsec Policy Agent** service which enforces IPsec policies.
- **Windows Explorer** for Windows 10 which can be used to manage the OS and check the integrity of Windows files and updates.
- The **Windows Trusted Installer** which installs updates to the Windows operating system.
- The **Key Isolation Service** which protects secret and private keys.

2.3.2 Physical Boundaries

Physically the TOE executes on x64 Intel-based processors. Refer to section 1.1 for the specific list of hardware used in the evaluation.

A set of devices may be attached as part of the TOE:

- Display Monitors
- Fixed Disk Drives (including disk drives and solid state drives)
- Removable Disk Drives (including USB storage)
- Network Adaptor
- Keyboard
- Mouse
- Printer
- Audio Adaptor
- CD-ROM Drive
- Smart Card Reader
- Trusted Platform Module (TPM) version 2.0

While this list of devices is larger than is needed to evaluate the requirements in the IPsec VPN Client protection profile, it is the same set of devices components as in the Mobile Device Fundamentals protection profile evaluation and the General Purpose Operating System Protection Profile evaluation for Windows 10. By using the same set of components for these evaluations, consumers can gain assurance by using both core OS capabilities and Mobile Device Fundamentals in combination.

2.4 TOE Security Services

This section summarizes the security services provided by the TOE:

- **Security Audit:** Windows has the ability to collect audit data, review audit logs, protect audit logs from overflow, and restrict access to audit logs. Audit information generated by the system

[4] AuthIP key exchange was not examined in the Common Criteria portion of this evaluation.

includes the date and time of the event, the user identity that caused the event to be generated, and other event specific data. Authorized administrators can review audit logs and have the ability to search and sort audit records. Authorized Administrators can also configure the audit system to include or exclude potentially auditable events to be audited based on a wide range of characteristics. In the context of this evaluation, the protection profile requirements cover generating audit events and selecting which events should be audited.

- **Cryptographic Support:** Windows provides FIPS validated cryptographic functions that support encryption/decryption, cryptographic signatures, cryptographic hashing, cryptographic key agreement (which is not studied in this evaluation), and random number generation. The TOE additionally provides support for public keys, credential management and certificate validation functions and provides support for the National Security Agency's Suite B cryptographic algorithms. Windows also provides extensive auditing support of cryptographic operations, the ability to replace cryptographic functions and random number generators with alternative implementations,[5] and a key isolation service designed to limit the potential exposure of secret and private keys. In addition to using cryptography for its own security functions, Windows offers access to the cryptographic support functions for user-mode and kernel-mode programs. Public key certificates generated and used by Windows authenticate users and machines as well as protect both user and system data in transit.
    - **IPsec:** Windows implements IPsec to provide protected, authenticated, confidential, and tamper-proof networking between two peer computers.
- **User Data Protection**: In the context of this evaluation Windows protects user data by means of protected network tunnel based on IPsec and zeroizes memory before it is allocated to a subject process.
- **Identification and Authentication**: In the context of this evaluation, Windows provides the ability to use, store, and protect X.509 certificates that are used for IPsec VPN sessions along with capability to use a pre-shared key.
- **Protection of the TOE Security Functions**: Windows provides a number of features to ensure the protection of TOE security functions. Windows protects against unauthorized data disclosure and modification by using a suite of Internet standard protocols including IPsec, IKE, and ISAKMP. Windows ensures process isolation security for all processes through private virtual address spaces, execution context, and security context. The Windows data structures defining process address space, execution context, memory protection, and security context are stored in protected kernel-mode memory. Windows includes self-testing features that ensure the integrity of executable program images and its cryptographic functions. Finally, Windows provides a trusted update mechanism to update Windows binaries itself.
- **Trusted Path for Communications**: Windows uses the IPsec suite of protocols to provide a Virtual Private Network Connection (VPN) between itself, acting as a VPN client, and a VPN gateway.

[5] This option is not included in the Windows Common Criteria evaluation.

- **Security Management:** Windows includes several functions to manage security policies. Policy management is controlled through a combination of access control, membership in administrator groups, and privileges.

3 Security Problem Definition

The security problem definition consists of the threats to security, organizational security policies, and usage assumptions as they relate to Windows. The assumptions, threats, and policies are copied from the Protection Profile for IPsec Virtual Private Network (VPN) Clients ("IPsec VPN PP").

3.1 Threats to Security

Table 1 presents known or presumed threats to protected resources that are addressed by Windows based on conformance to the IPsec VPN Client PP.

Table 1 IPsec VPN Client PP Threats Addressed by Windows

Threat	Description
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.

3.2 Organizational Security Policies

An organizational security policy is a set of rules or procedures imposed by an organization upon its operations to protect its sensitive data and IT assets. **Table 2** describes organizational security policies which are necessary for conformance to the protection profile.

Table 2 Organizational Security Policies

Security Policy	Description
[None]	There are no Organizational Security Policies for the protection profile.

3.3 Secure Usage Assumptions

Table 3 describes the core security aspects of the environment in which Windows is intended to be used. It includes information about the physical, personnel, procedural, and connectivity aspects of the environment.

The following specific conditions are assumed to exist in an environment where the TOE is employed in order to conform to the protection profile:

Table 3 Secure Usage Assumptions

Assumption	Description
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.

4 Security Objectives

This section defines the security objectives for Windows and its supporting environment. Security objectives, categorized as either TOE security objectives or objectives by the supporting environment, reflect the stated intent to counter identified threats, comply with any organizational security policies identified, or address identified assumptions. All of the identified threats, organizational policies, and assumptions are addressed under one of the categories below.

4.1 TOE Security Objectives

Table 4 describes the security objectives for Windows which are needed to comply with the IPsec VPN Client PP.

Table 4 Security Objectives for the TOE

Security Objective	Source
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.

4.2 Security Objectives for the Operational Environment

The TOE is assumed to be complete and self-contained and, as such, is not dependent upon any other products to perform properly. However, certain objectives with respect to the general operating environment must be met. **Table 5** describes the security objectives for the operational environment as specified in the protection profile.

Table 5 Security Objectives for the Operational Environment

Environment 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.

5 Security Requirements

The section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) for the TOE. The requirements in this section have been drawn from the Protection Profile for IPsec Virtual Private Network (VPN) Clients, Version 1.4, October 21, 2013, or the Common Criteria.

Conventions:

Where requirements are drawn from the protection profile, the requirements are copied verbatim, except for some changes to required identifiers to match the iteration convention of this document, from that protection profile and only operations performed in this security target are identified.

The extended requirements, extended component definitions and extended requirement conventions in this security target are drawn from the protection profile; the security target reuses the conventions from the protection profile which include the use of the word "Extended" and the "_EXT" identifier to denote extended functional requirements. The security target assumes that the protection profile correctly defines the extended components and so they are not reproduced in the security target.

Where applicable the following conventions are used to identify operations:

- **Iteration**: Iterated requirements (components and elements) are identified with letter following the base component identifier. For example, iterations of FMT_MOF.1 are identified in a manner similar to FMT_MOF.1(Audit) (for the component) and FCS_COP.1.1(Audit) (for the elements).
- **Assignment**: Assignments are identified in brackets and bold (e.g., **[assigned value]**).
- **Selection**: Selections are identified in brackets, bold, and italics (e.g., ***[selected value]***).
    - Assignments within selections are identified using the previous conventions, except that the assigned value would also be italicized and extra brackets would occur (e.g., ***[selected value [assigned value]]***).
- **Refinement**: Refinements are identified using bold text (e.g., **added text**) for additions and strike-through text (e.g., ~~deleted text~~) for deletions.

The security target uses footnotes to describe small differences in product capabilities in both section 5 (Security Requirements) and section 6 (TOE Summary Specification).

5.1 TOE Security Functional Requirements

This section specifies the SFRs for the TOE.

Table 6 TOE Security Functional Requirements

Requirement Class	Requirement Component
Security Audit (FAU)	Audit Data Generation (FAU_GEN.1)
	Selective Audit (FAU_SEL.1)
	Cryptographic Key Generation for Asymmetric Keys (FCS_CKM.1(ASYM))
Cryptographic Support (FCS)	Cryptographic Key Generation for IKE Asymmetric Keys (FCS_CKM.1(IKE))
	Extended: Cryptographic Key Storage (FCS_CKM_EXT.2)

	Extended: Cryptographic Key Zeroization (FCS_CKM_EXT.4)
	Cryptographic Operation for Data Encryption/Decryption (FCS_COP.1(SYM))
	Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))
	Cryptographic Operation for Hashing (FCS_COP.1(HASH))
	Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))
	Extended: IPsec Communications (FCS_IPSEC_EXT.1)
	Extended: Random Bit Generation (FCS_RBG_EXT.1)
User Data Protection (FDP)	Extended: Subset Information Flow Control (FDP_IFC_EXT.1)
	Residual Information Protection (FDP_RIP.2)
Identification & Authentication (FIA)	Extended: Pre-Shared Key Composition (FIA_PSK_EXT.1)
	Extended: X509 Certificates (FIA_X509_EXT.1)
	Extended: X509 Certificate Use and Management (FIA_X509_EXT.2)
Security Management (FMT)	Specification of Management Functions (FMT_SMF.1(TOE))
	Specification of Management Functions (FMT_SMF.1(MGMT))
Protection of the TSF (FPT)	Extended: TSF Self Test (FPT_TST_EXT.1)
	Extended: Trusted Update (FPT_TUD_EXT.1)
Trusted Path/Channels (FTP)	Trusted Channel (FTP_ITC.1)

5.1.1 Security Audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)[6]

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 Table **7** ~~3~~].

FAU_GEN.1.2 The [***TOE***] 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 **7** ~~2~~ below].

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 for authentication keys.	No additional information

[6] This protection profile requirement was modified as part of NIAP Technical Decision 42.

FCS_CKM_EXT.2	None.	None.
FCS_CKM_EXT.4	None.	None.
FCS_COP.1(SYM)	None.	None.
FCS_COP.1(SIGN)	None.	None.
FCS_COP.1(HASH)	None.	None.
FCS_COP.1(HMAC)	None.	None.
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	None.	None.
FDP_IFC_EXT.1	Failure to establish exclusive tunnel.	None.
FDP_RIP.2	None.	
FIA_PSK_EXT.1	Failure of the randomization process.	None.
FIA_X509_EXT.1	Failure of the X.509 certificate validation	Reason for failure of validation.
FIA_X509_EXT.2	[if one were required] Failure of the path validation of the X.509 certificate	Reason for failure of path validation.
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.

Table 7 Auditable Events

5.1.1.2 *Selective Audit (FAU_SEL.1)*

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) [**Object, subject or user identity and host**].

5.1.2 Cryptographic Support (FCS)

5.1.2.1 Cryptographic Key Generation for Asymmetric Keys (FCS_CKM.1(ASYM))

Application Note: FCS_CKM.1(ASYM)) corresponds to FCS_CKM.1(1) in the IPsec VPN Client protection profile.

FCS_CKM.1.1(ASYM)) The [***TOE***] 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 [***no other curves***][7] (as defined in FIPS PUB 186-4, "Digital Signature Standard")
- [***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.

5.1.2.2 Cryptographic Key Generation for IKE Asymmetric Keys (FCS_CKM.1(IKE))

Application Note: FCS_CKM.1(IKE)) corresponds to FCS_CKM.1(2) in the IPsec VPN Client protection profile.

FCS_CKM.1.1(IKE)) The [***TOE***] shall generate asymmetric cryptographic keys used for IKE peer authentication in accordance with a:
[
- ***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 [no other curves];***[8]

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

5.1.2.3 Extended: Cryptographic Key Storage (FCS_CKM_EXT.2)

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

[7] While Windows implements the P-521 curve, it does not expose an IPsec cryptosuite which leverages it
[8] See previous comment about the P-521 curve.

5.1.2.4 Extended: Cryptographic Key Zeroization (FCS_CKM_EXT.4)

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

5.1.2.5 Cryptographic Operation for Data Encryption/Decryption (FCS_COP.1(SYM))

Application Note: FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the IPsec VPN Client protection profile.

FCS_COP.1.1(SYM)) The [***TOE***] 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.

5.1.2.6 Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))

Application Note: FCS_COP.1(SIGN) corresponds to FCS_COP.1(2) in the IPsec VPN Client protection profile.

FCS_COP.1.1(SIGN)) The [***TOE***] shall perform cryptographic signature services in accordance with a specified cryptographic algorithm:

- [***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 [no other curve]***] [9]

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

5.1.2.7 Cryptographic Operation for Hashing (FCS_COP.1(HASH))

Application Note: FCS_COP.1(HASH) corresponds to FCS_COP.1(3) in the IPsec VPN Client protection profile.

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

5.1.2.8 Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))

Application Note: FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the IPsec VPN Client protection profile.

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

[9] See previous comment about the P-521 curve.

5.1.2.9 Extended: IPsec Communications (FCS_IPSEC_EXT.1)

FCS_IPSEC_EXT.1.1	The [***TOE***] shall implement the IPsec architecture as specified in RFC 4301.
FCS_IPSEC_EXT.1.2	The [***TOE***] shall implement [***tunnel mode, transport mode***].
FCS_IPSEC_EXT.1.3	The [***TOE***] shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.
FCS_IPSEC_EXT.1.4	The [***TOE***] shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, [***AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC***].
FCS_IPSEC_EXT.1.5	The [***TOE***] shall implement the protocol: [***IKEv1 as defined in RFCs 2407, 2408, 2409, RFC 4109, [RFC 4304 for extended sequence numbers], and [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 [RFC 4868 for hash functions]***].
FCS_IPSEC_EXT.1.6	The [***TOE***] shall ensure the encrypted payload in the [***IKEv1, IKEv2***] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [***no other algorithm***].
FCS_IPSEC_EXT.1.7	The [***TOE***] shall ensure that IKEv1 Phase 1 exchanges use only main mode.
FCS_IPSEC_EXT.1.8	The [***TOE***] shall ensure that [***IKEv2 SA lifetimes can be configured by [an Administrator, VPN Gateway] based on [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 [an Administrator, VPN Gateway] based on [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]***].
FCS_IPSEC_EXT.1.9	The [***TOE***] shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [**224, 256, 384**] bits.
FCS_IPSEC_EXT.1.10	The [***TOE***] shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**256**] .
FCS_IPSEC_EXT.1.11	The [***TOE***] shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), and [***20 (384-bit Random ECP)***].
FCS_IPSEC_EXT.1.12	The [***TOE***] shall ensure that all IKE protocols perform peer authentication using a [***RSA, ECDSA***] that use X.509v3 certificates that conform to RFC 4945 and [***Pre-shared Keys***].
FCS_IPSEC_EXT.1.13	The [***TOE***] shall support peer identifiers of the following types: [***IP address, Fully Qualified Domain Name (FQDN), user FQDN***] and [***no other reference identifier type***]. The TSF shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.[10]
FCS_IPSEC_EXT.1.14	The [***TOE***] 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 [***IKEv1 Phase 1, IKEv2 IKE_SA***] connection is greater than or equal to the

[10] This protection profile requirement was modified as part of NIAP Technical Decision 37.

strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 2, IKEv2 CHILD_SA***] connection.

5.1.2.10 Extended: Random Bit Generation (FCS_RBG_EXT.1)[11]

FCS_RBG_EXT.1.1	The [***TOE***] shall perform all deterministic random bit generation services in accordance with [***NIST Special Publication 800-90A using [CTR_DRBG (AES)]***].
FCS_RBG_EXT.1.2	The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***a software-based noise source, a platform-based RBG***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

5.1.3 User Data Protection (FDP)

5.1.3.1 Extended: Subset Information Flow Control (FDP_IFC_EXT.1)

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.

5.1.3.2 Residual Information Protection (FDP_RIP.2)

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

5.1.4 Identification and Authentication (FIA)

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

FIA_PSK_EXT.1.1 The [***TOE***] shall be able to use pre-shared keys for IPsec.

FIA_PSK_EXT.1.2 The [***TOE***] shall be able to accept text-based preshared keys that:

- are 22 characters and [***[less than 10,000 characters]***];
- composed of any combination of upper and lower case letters, numbers, and special characters (that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”).

FIA_PSK_EXT.1.3 The [***TOE***] shall [***condition the text-based pre-shared keys by using [No conditioning]***].

5.1.4.2 Extended: X509 Certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1 The [***TOE***] 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 [***the Online Certificate Status Protocol (OCSP) as specified in RFC 2560, a Certificate Revocation List (CRL) as specified in RFC 5759***].

[11] This protection profile requirement was modified as part of NIAP Technical Decision 79.

- 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 [***trusted updates, integrity verification***] shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).

FIA_X509_EXT.1.2 The [***TOE***] 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.

5.1.4.3 Extended: X509 Certificate Use and Management (FIA_X509_EXT.2)

FIA_X509_EXT.2.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec exchanges, and [***digital signatures for FPT_TUD_EXT.1, integrity checks for FPT_TST_EXT.1.2***].

FIA_X509_EXT.2.2 When a connection to determine the validity of a certificate cannot be established, the [***TOE***] shall [***not accept the certificate***].

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

5.1.5 Security Management (FMT)

5.1.5.1 Specification of Management Functions (FMT_SMF.1(TOE))[12]

FMT_SMF.1.1(TOE) 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,
- [**none**].

5.1.5.2 Specification of Management Functions (FMT_SMF.1(MGMT))[13]

FMT_SMF.1.1 The [***TOE, 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,
- [***no other actions***].

[12] FMT_SMF.1(TOE) corresponds to the FMT_SMF.1 requirement in section 4.1.1 of the protection profile.
[13] FMT_SMF.1(MGMT) corresponds to the FMT_SMF.1 requirement in section 4.2.4 of the protection profile.

5.1.6 Protection of the TSF (FPT)

5.1.6.1 Extended: TSF Self Test (FPT_TST_EXT.1)

FPT_TST_EXT.1.1	The [*TOE*] 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 [*TOE*] shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the [**a digital signature as described in FCS_COP.1(SIGN)**].

5.1.6.2 Extended: Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1	The [*TOE*] shall provide the ability to query the current version of the TOE firmware/software.
FPT_TUD_EXT.1.2	The [*TOE*] shall provide the ability to initiate updates to TOE firmware/software.
FPT_TUD_EXT.1.3	The [*TOE*] shall provide a means to verify firmware/software updates to the TOE using a digital signature mechanism and [***no other functions***] prior to installing those updates.

5.1.7 Trusted Path / Channels (FTP)

5.1.7.1 Trusted Channel (FTP_ITC.1)

FTP_ITC.1.1	The [*TOE*] 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 [*TOE*] shall permit the TSF to initiate communication via the trusted channel.
FTP_ITC.1.3	The [*TOE*] shall initiate communication via the trusted channel for all traffic traversing that connection.

5.2 TOE Security Assurance Requirements

5.2.1 CC Part 3 Assurance Requirements

The following table is the collection of CC Part 3 assurance requirements from the Protection Profile for IPsec Virtual Private Network (VPN) Clients.

Requirement Class	Requirement Component
ADV: Design	ADV_FSP.1: Basic functional specification
AGD: Guidance Documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle Support	ALC_CMC.1: Labeling of the TOE
	ALC_CMS.1: TOE CM coverage
ATE: Testing	ATE_IND.1: Independent testing - sample

AVA: Vulnerability Assessment	AVA_VAN.1: Vulnerability survey

5.2.2 IPsec VPN Client PP Assurance Activities

This section copies the assurance activities from the protection profile in order to ease reading and comparisons between the protection profile and the security target.

5.2.2.1 Security Audit (FAU)

5.2.2.1.1 Audit Data Generation (FAU_GEN.1)

5.2.2.1.1.1 FAU_GEN.1.1

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.

5.2.2.1.1.2 FAU_GEN.1.2

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

5.2.2.1.2 Selective Audit (FAU_SEL.1)

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.

5.2.2.2 Cryptographic Support (FCS)

5.2.2.2.1 Cryptographic Key Generation for Asymmetric Keys (FCS_CKM.1(ASYM))[14]

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.

[14] FCS_CKM.1(ASYM) corresponds to FCS_CKM.1(1) in the IPsec VPN Client protection profile.

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:
   1. Provable primes
   2. 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.

<u>SP800-56A Key Establishment Schemes</u>

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.

<u>SP800-56B Key Establishment Schemes</u>

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.

5.2.2.2.2 Cryptographic Key Generation for IKE Asymmetric Keys (FCS_CKM.1(IKE))[15]

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.

[15] FCS_CKM.1(IKE) corresponds to FCS_CKM.1(2) in the IPsec VPN Client protection profile.

5.2.2.2.3 Extended: Cryptographic Key Storage (FCS_CKM_EXT.2)

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.

5.2.2.2.4 Extended: Cryptographic Key Zeroization (FCS_CKM_EXT.4)

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 protocol); 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.

5.2.2.2.5 Cryptographic Operation for Data Encryption/Decryption (FCS_COP.1(SYM))[16]

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.

<u>**AES-CBC Tests**</u>

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

[16] FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the IPsec VPN Client protection profile.

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 allzeros 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 5tuples 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.

5.2.2.2.6 Cryptographic Operation for Signature Algorithms (FCS_COP.1(SIGN))[17]

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.

<u>Key Generation:</u>

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:

3. Random Primes:
8. Provable primes
9. Probable primes
4. 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

[17] FCS_COP.1(SIGN) corresponds to FCS_COP.1(2) in the IPsec VPN Client protection profile.

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.

<u>ECDSA Key Generation Tests</u>

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.

<u>ECDSA Algorithm Tests</u>

ECDSA 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.

<u>RSA Signature Algorithm Tests</u>

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.

5.2.2.2.7 Cryptographic Operation for Hashing (FCS_COP.1(HASH))[18]

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.

<u>Short Messages Test - Bit-oriented Mode</u>

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.

<u>Short Messages Test - Byte-oriented Mode</u>

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.

<u>Selected Long Messages Test - Bit-oriented Mode</u>

[18] FCS_CKM.1(HASH) corresponds to FCS_CKM.1(3) in the IPsec VPN Client protection profile.

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.

5.2.2.2.8 Cryptographic Operation for Keyed Hash Algorithms (FCS_COP.1(HMAC))[19]

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

[19] FCS_CKM.1(HMAC) corresponds to FCS_CKM.1(4) in the IPsec VPN Client protection profile.

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.

5.2.2.2.9 Extended: IPsec Communications (FCS_IPSEC_EXT.1)

5.2.2.2.9.1 FCS_IPSEC_EXT.1.1

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.

5.2.2.2.9.2 FCS_IPSEC_EXT.1.2

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.

5.2.2.2.9.3 FCS_IPSEC_EXT.1.3

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.

5.2.2.2.9.4 FCS_IPSEC_EXT.1.4

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 AESCBC-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 AESCBC-256, the TOE/platform is configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.

5.2.2.2.9.5 FCS_IPSEC_EXT.1.5

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.

5.2.2.2.9.6 FCS_IPSEC_EXT.1.6

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.

5.2.2.2.9.7 FCS_IPSEC_EXT.1.7

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.

5.2.2.2.9.8 FCS_IPSEC_EXT.1.8

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.

5.2.2.2.9.9 FCS_IPSEC_EXT.1.9

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.

5.2.2.2.9.10 FCS_IPSEC_EXT.1.10

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.

5.2.2.2.9.11 FCS_IPSEC_EXT.1.11

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.

5.2.2.2.9.12 FCS_IPSEC_EXT.1.12[20]

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.

[20] This protection profile assurance activity was modified as part of NIAP Technical Decision 53.

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.

5.2.2.2.9.13 FCS_IPSEC_EXT.1.13[21]
TSS

The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN), and, if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.

GUIDANCE

The evaluator shall ensure that the operational guidance includes the configuration of the reference identifier(s) for the peer.

TEST

For each supported identifier type (excluding DNs), the evaluator shall repeat the following tests:

Test 1: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 2: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKE authentication fails.

The following tests are conditional:

Test 3: (conditional) If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented certificate but to match the Common Name in the peer's presented certificate, and verify that the IKE authentication fails.

Test 4: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKE authentication fails.

Test 5: (conditional) If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the

[21] This protection profile assurance activity was modified as part of NIAP Technical Decision 37.

presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.

Test 6: (conditional) If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.

5.2.2.2.9.14 FCS_IPSEC_EXT.1.14

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.

5.2.2.2.10 Extended: Random Bit Generation (FCS_RBG_EXT.1)

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 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 80090).

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.

5.2.2.3 User Data Protection (FDP)

5.2.2.3.1 Extended: Subset Information Flow Control (FDP_IFC_EXT.1)

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.

5.2.2.3.2 Residual Information Protection (FDP_RIP.2)

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.

5.2.2.4 Identification and Authentication (FIA)

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

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.

5.2.2.4.2 Extended: X509 Certificates (FIA_X509_EXT.1)

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.

5.2.2.4.3 Extended: X509 Certificate Use and Management (FIA_X509_EXT.2)

5.2.2.4.3.1 FIA_X509_EXT.2.1

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.

5.2.2.4.3.2 FIA_X509_EXT.2.2

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.

5.2.2.4.3.3 FIA_X509_EXT.2.3

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

5.2.2.5 Security Management (FMT)

5.2.2.5.1 Specification of Management Functions (FMT_SMF.1)

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.

GUIDANCE

The evaluator shall ensure that the operational guidance instructs the administrator on configuring the reference identifier of the peer.

TEST

The evaluator follows this guidance in the performance of the assurance activities for the appropriate FCS_IPSEC or FIA_X509 requirement.

5.2.2.5.2 Specification of Management Functions (FMT_SMF.1)

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.

5.2.2.6 Protection of the TSF (FPT)

5.2.2.6.1 Extended: TSF Self Test (FPT_TST_EXT.1)

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.

5.2.2.6.2 Extended: Trusted Update (FPT_TUD_EXT.1)

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 evaluator

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.

5.2.2.7 Trusted Path / Channels (FTP)

5.2.2.7.1 Trusted Channel (FTP_ITC.1)

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.

6 TOE Summary Specification (TSS)

This chapter describes the Windows security functions which satisfy the security functional requirements of the protection profile. The TOE also includes additional relevant security functions which are also described in the following sections, as well as a mapping to the security functional requirements satisfied by the TOE.

6.1 TOE Security Functions

This section presents the TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs). The TOE performs the following security functions:

- Audit
- Cryptographic Support
- User Data Protection
- Identification and Authentication
- Security Management
- Protection of the TSF
- Trusted Path / Channels

6.2 Audit

The TOE Audit security function performs:

- Audit Collection
- Selective Audit

6.2.1 Audit Collection

The Windows Event Log service creates the security event log, which contains security relevant audit records collected on a system, along with other event logs which are also registered by other audit entry providers. The Local Security Authority (LSA) server collects audit events from all other parts of the TSF and forwards them to the Windows Event Log service which will place the event into the log for the appropriate provider. While there is no size limit for a single audit record, the authorized administrator can specify a limit for the size of each event log. For each audit event, the Windows Event Log service stores the following data in each audit entry:

Field in Audit Entry	Description
Date	The date the event occurred.
Time	The time the event occurred.
User	The security identifier (SID) that represents the user on whose behalf the event occurred.
Event ID	A unique number within the audit category that identifies the specific audit event.
Source	The Windows component that generated the audit event.

Outcome	Indicates whether the security audit event recorded is the result of a successful or failed attempt to perform the action.
Category	The type of the event defined by the event source.

Table 8 Standard Fields in a Windows Audit Entry

The LSA service defines the following categories for audit events in the security log:

- System,
- Logon / Logoff
- Object Access
- Directory Service Access
- Privilege Use
- Detailed Process Tracking
- Policy Change
- Account Management
- Account Logon

Each audit entry may also contain category-specific data that is contained in the body of the entry as described below:

- For the System Category, the audit entry includes information relating to the system such as the time the audit trail was cleared, start or shutdown of the audit function, and startup and shutdown of Windows. Furthermore, the specific cryptographic operation is identified when such operations are audited.
- For the Logon and Account Logon Category, the audit entry includes the reason the attempted logon failed.
- For the Object Access and the Directory Service Access Category, the audit entry includes the object name and the desired access requested.
- For the Privilege Use Category, the audit entry identifies the privilege.
- For the Detailed Process Tracking Category, the audit event includes the process identifier.
- For the Policy Change and Account Management Category, the audit event includes the new values of the policy or account attributes.
- For the Account Logon Category, the audit event includes the logon type that indicates the source of the logon attempt as one of the following types in the audit record:
  - Interactive (local logon)
  - Network (logon from the network)
  - Service (logon as a service)
  - Batch (logon as a batch job)
  - Unlock (for Unlock screen saver)
  - Network_ClearText (for anonymous authentication to IIS)

There are two places within the TSF where security audit events are collected. Inside the kernel, the Security Reference Monitor (SRM), a part of the NT Executive, is responsible for generation of all audit entries for the object access, privilege use, and detailed process tracking event categories. Windows components can request the SRM to generate an audit record and supply all of the elements in the audit record except for the system time, which the Executive provides. With one exception, audit events for the other event categories are generated by various services that either co-exist in the LSA server or call, with the SeAuditPrivilege privilege, the Authz Report Audit interfaces implemented in the LSA Policy subcomponent. The exception is that the Event Log Service itself records an event record when the security log is cleared and when the security log exceeds the warning level configured by the authorized administrator.

The LSA server maintains an audit policy in its database that determines which categories of events are actually collected. Defining and modifying the audit policy is restricted to the authorized administrator. The authorized administrator can select events to be audited by selecting the category or categories to be audited. An authorized administrator can individually select each category. Those services in the security process determine the current audit policy via direct local function calls. The only other TSF component that uses the audit policy is the SRM in order to record object access, privilege use, and detailed tracking audit. LSA and the SRM share a private local connection port, which is used to pass the audit policy to the SRM. When an authorized administrator changes the audit policy, the LSA updates its database and notifies the SRM. The SRM receives a control flag indicating if auditing is enabled and a data structure indicating that the events in particular categories to audit.

In addition to the system-wide audit policy configuration, it is possible to define a per-user audit policy using auditpol.exe. This allows individual audit categories (of success or failure) to be enabled or disabled on a per user basis.[22]   The per-user audit policy refines the system-wide audit policy with a more precise definition of the audit policy for which events will be audited for a specific user.

Within each category, auditing can be performed based on success, failure, or both. For object access events, auditing can be further controlled based on user/group identify and access rights using System Access Control Lists (SACLs). SACLs are associated with objects and indicate whether or not auditing for a specific object, or object attribute, is enabled.

The TSF is capable of generating the audit events associated with each audit category, as described in the Description column of **Table 9 Audit Event Categories**. The auditable events associated with each category capture the events listed in section **5.1.1.1**. For each category, the associated audit events (listed in **5.1.1.1**) for each of the requirements in the FAU_GEN Required Events column of **Table 9** are captured.

Category	Description	FAU_GEN Required Events
System	Audit attempts that affect security of the entire system such as clearing the audit trail.	FCS_CKM.1*, FDP_IFC_EXT.1
Object Access	Audit attempts to access user objects, such as files.	None for the IPsec VPN Client PP.

[22] Windows will prevent a local administrator from disabling auditing for local administrator accounts. If an administrator can bypass auditing, they can avoid accountability for such actions as exfiltrating files without authorization.

Privilege Use	Audits attempts to use security relevant privileges. Security relevant privileges are those privileges that are related to the TSFs and can be assigned in the evaluated configuration.	None for the IPsec VPN Client PP.
Detailed Process Tracking	Audit subject-tracking events, including program activation, handle duplication, indirect access to an object, and process exit.	None for the IPsec VPN Client PP.
Policy Change	Audit attempts to change security policy settings such as the audit policy and privilege assignment.	FAU_SEL.1
Account Management	Audit attempts to create, delete, or change user or group accounts and changes to their attributes.	None for the IPsec VPN Client PP.
Directory Service Access	Audit access to directory service objects and associated properties.	None for the IPsec VPN Client PP.
Logon	Audit attempts to logon or logoff the system, attempts to make a network connection.	None for the IPsec VPN Client PP.
Account Logon	Audit when a DC receives a logon request.	None for the IPsec VPN Client PP.
IPsec Main Mode	Audit records related to IPsec protocol actions	FCS_IPSEC_EXT.1, FTP_ITC.1
Certificate validation	Failure to validate a X.509 v3 certificate	FIA_X509_EXT.1
Product update	Audit records related to product update or installation	FPT_TUD_EXT.1

Table 9 Audit Event Categories

6.2.2 Selective Audit

The authorized administrator has the ability to select events to be audited based upon object identity, user identity, computer (host identity), type (category), and outcome (success or failure) of the event. Selecting the set of events that will be audited can be on a per-machine basis by using tools such as auditpol.exe and wevtutil.exe, or using group policies to audit sets of machines (i.e. auditing based on the host identity).

6.2.3 SFR Mapping

The **Audit** function satisfies the following SFRs:

- **FAU_GEN.1**: The TOE audit collection is capable of generating audit events for items identified in section **5.1.1.1**. For each audit event the TSF records the date, time, user Security Identifier (SID) or name, logon type (for logon audit records), event ID, source, type, and category.

- **FAU_SEL.1**: The TSF provides the ability for the authorized administrator to select the events to be audited based upon object identity, user identity, workstation (host identity), event type, and success or failure of the event.

6.3 Cryptographic Support

Cryptography API: Next Generation (CNG) API is designed to be extensible at many levels and agnostic to cryptographic algorithm suites. An important feature of CNG is its native implementation of the Suite B algorithms, including algorithms for AES (128, 192, 256 key sizes)[23], the SHA-1 and SHA-2 family (SHA-256, SHA-384 and SHA-512) of hashing algorithms, elliptic curve Diffie Hellman (ECDH), and elliptical curve DSA (ECDSA) over the NIST-standard prime curves P-256, P-384, and P-521.

Protocols such as the Internet Key Exchange (IKE), and Transport Layer Security (TLS), make use of elliptic curve Diffie-Hellman (ECDH) included in Suite B as well as hashing functions.

Deterministic random bit generation (DRBG) is implemented in accordance with NIST Special Publication 800-90. Windows generates random bits by taking the output of a cascade of two SP800-90 AES-256 counter mode based DRBGs in kernel-mode and four cascaded SP800-90 AES-256 DRBGs in user-mode; programmatic callers can choose to obtain either 128 or 256 bits from the RBG which is seeded from the Windows entropy pool. The entropy pool is populated using the following values:

- An initial entropy value from a seed file provided to the Windows OS Loader at boot time (512 bits of entropy).
- A calculated value based on the high-resolution CPU cycle counter which fires after every 1024 interrupts (a continuous source providing 16384 bits of entropy).
- Random values gathered periodically from the Trusted Platform Module (TPM), (320 bits of entropy on boot, 384 bits thereafter).
- Random values gathered periodically by calling the RDRAND CPU instruction, (256 bits of entropy).

The main source of entropy in the system is the CPU cycle counter which tracks hardware interrupts. This is a sufficient health test; if the computer were not accumulating hardware and software interrupts every processor clock cycle it would not be running and therefore there would be no need for random bit generation. In the same manner, a failure of the TPM chip or processor would be a critical error that halts the computer. In addition, when the user follows the CC administrative guidance, which includes operating Windows in the FIPS validated mode, it will run FIPS 140 AES-256 Counter Mode DBRG Known Answer Tests (instantiate, generate) on start-up. Windows always runs the SP 800-90-mandated self-tests for AES-CTR-DRBG during a reseed.[24]

[23] Note that the 192-bit key size is not used by Windows but is available to developers.
[24] Running Windows in FIPS validated mode is required according to the administrative guidance.

Each entropy source is independent of the other sources and does not depend on time. The CPU cycle counter inputs vary by environmental conditions such as data received on a network interface card, key presses on a keyboard, mouse movement and clicks, and touch input.

The TSF defends against tampering of the random number generation (RNG) / pseudorandom number generation (PRNG) sources by encapsulating its use in Kernel Security Device Driver. The interface for the Windows random number generator is BCryptGenRandom.

The CNG provider for random number generation is the AES_CTR_DRBG, when Windows requires the use of a salt it uses the Windows RBG.

The encryption and decryption operations are performed by independent modules, known as Cryptographic Service Providers (CSPs). Windows generates symmetric keys (AES keys) using the FIPS Approved random number generator.

In addition to encryption and decryption services, the TSF provides other cryptographic operations such as hashing and digital signatures. Hashing is used by other FIPS Approved algorithms implemented in Windows (the hashed message authentication code, RSA, DSA, and EC DSA signature services, Diffie-Hellman and elliptic curve Diffie-Hellman key agreement, and random bit generation). When Windows needs to establish an RSA-based shared secret key it can act both as a sender or recipient, any decryption errors which occur during key establishment are presented to the user at a highly abstracted level, such as a failure to connect.

The hash-based message authentication code functions (HMAC) are based on SHA-1, SHA-256, and SHA-384, have the following characteristics:

HMAC Algorithm	Hash function Used	Block Size	Output MAC Length	Key Length / Key Size
HMAC-SHA-1	SHA-1	512 bits	20 bytes	The key size is 10-63 bytes when the key size is less than the block size and the key size is 65 to 1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.
HMAC-SHA-256	SHA-256	512 bits	32 bytes	Same as HMAC-SHA-1
HMAC-SHA-384	SHA-384	1024 bits	48 bytes	The key size is 24-127 bytes when the key size is less than the block size and the key size is 129-1024 bytes when the key size is greater than the block size. The key size may also equal the block size. The key size is variable.

Table 10 HMAC Characteristics

Cryptographic Operation	Standard	Evaluation Method
Encryption/Decryption	FIPS 197 AES For CBC and GCM modes	NIST CAVP #3630, #3629
Digital signature	FIPS 186-4 RSA	NIST CAVP #1889, #1888, #1887, #1871

Digital signature	FIPS 186-4 DSA	NIST CAVP #1024
Digital signature	FIPS 186-4 ECDSA	NIST CAVP #760
Hashing	FIPS 180-4 SHA-2	NIST CAVP #3048, #3047
Keyed-Hash Message Authentication Code	FIPS 198-2 HMAC	NIST CAVP #2381
Random number generation	NIST SP 800-90 CTR_DRBG	NIST CAVP #955
Key agreement	NIST SP 800-56A ECDH	NIST CAVP #72
Key establishment	NIST SP 800-56B	NIST CVL #663
Key-based key derivation	NIST SP 800-108	NIST CAVP #72
IKEv1	NIST SP 800-135	NIST CVL #664
IKEv2	NIST SP 800-135	NIST CVL #664

Table 11 Cryptographic Standards and Evaluation Methods

The TSF includes a key isolation service designed specifically to host secret and private keys in a protected process to mitigate tampering or access to sensitive key materials. The TSF performs a key error detection check on each transfer of key (internal and intermediate transfers). The TSF prevents archiving of expired (private) signature keys. The TSF destroys non-persistent cryptographic keys by a single direct overwrite consisting of zeros.

Security Relevant Data Item	Description
Symmetric encryption/decryption keys	Keys used for AES (FIPS 197) encryption/decryption for IPsec ESP.
HMAC keys	Keys used for HMAC-SHA1, HMAC-SHA256, HMAC-SHA384, and HMAC-SHA512 (FIPS 198-1)
Asymmetric ECDSA Public Keys	Keys used for the verification of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric ECDSA Private Keys	Keys used for the calculation of ECDSA digital signatures (FIPS 186-4) for IPsec traffic and peer authentication.
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures (FIPS 186-4) for IPsec and signed product updates.
Asymmetric RSA Private Keys	Keys used for the calculation of RSA digital signatures (FIPS 186-4) for IPsec.
AES-CTR DRBG Seed	A secret value maintained internal to the module that provides the seed material for AES-CTR DRBG output.
AES-CTR DRBG Entropy Input	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output (SP 800-90).
AES-CTR DRBG V	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output (SP 800-90).
AES-CTR DRBG Key	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output (SP 800-90).

DH Private and Public values	Private and public values used for Diffie-Hellman key establishment.
ECDH Private and Public values	Private and public values used for EC Diffie-Hellman key establishment.

Table 12 Types of Keys and Cryptographic Materia Used by Windows

These keys and critical security parameters are zeroized, using the process described above, when they are no longer needed.

6.3.1 IPsec

The Windows IPsec implementation conforms to RFC 4301, Security Architecture for the Internet Protocol. This is documented publicly in the Windows protocol documentation at section 7.5.1 IPsec Overview and covers Windows 8, Windows RT, and Server 2012.[25]

Windows implements both RFCS 2409, Internet Key Exchange (IKEv1), and RFC 4306, Internet Key Exchange version 2, (IKEv2).[26] Windows IPsec supports both tunnel mode and transport mode and provides an option for NAT transversal (reference: section 7.5.5, IPsec Encapsulations).[27] The RAS VPN interface uses tunnel mode only.

The Windows IPsec implementation includes a security policy database (SPD), which states how Windows should process network packets. The SPD uses the traffic source, destination and transport protocol to determine if a packet should be transmitted or received, blocked, or protected with IPsec, (reference: 7.5.3, Security Policy Database Structure), based on firewall processing rules.[28] These rules are described in Understanding Firewall Rules and section 4 of *Microsoft Windows Common Criteria Supplemental Admin Guidance for IPsec VPN Clients*. In order to prevent unsolicited inbound traffic, an authorized administrator does not need to define a final catch-all rule which will discard a network packet when no other rules in the SPD apply because Windows will discard the packet. The security policy database also includes configuration settings to limit the time and number of sessions before a new key needs to be generated.

Windows 10 implements AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256 as encryption algorithms for the encapsulating security payload (ESP) (reference: section 6, Appendix A, Product Behavior).[29] However only AES-CBC-128 and AES-CBC-256 can be used for IKEv1 and IKEv2 to protect the encrypted payload. The resulting potential strength of the symmetric key will be 128 or 256 bits of security depending on whether the IPsec VPN client and IPsec VPN server agreed to use a 128 or 256 AES symmetric key to protect the network traffic. Windows implements HMAC-SHA1, HMAC-SHA-256 and HMAC-SHA-384[30] as authentication algorithms for key exchange as well as Diffie-Hellman Groups

[25] Also available as [MS-WSO], *Windows System Overview*, page 43 for offline reading.
[26] [MS-IKEE], *Internet Key Exchange Protocol Extensions*, page 8.
[27] [MS-WSO], page 45.
[28] [MS-WPO], page 44.
[29] [MS-IKEE], pages 74 – 75.
[30] Windows truncates the HMAC output as described in RFC 4868 for HMAC-SHA-256 and HMAC-SHA-384 and for HMAC-SHA1-96 as described in RFC 2404.

14, 19, and 20 (reference: section 6, Appendix A, Product Behavior).[31] The IPsec VPN client will propose a cryptosuite to the IPsec VPN server; if the server responds with a cryptosuite that the client supports, the client will use the server's proposed cryptosuite instead. If the IPsec VPN client and server cannot agree on a cryptosuite, either side may terminate the connection attempt.

In order to prevent security being reduced while transitioning from IKE Phase 1 / IKEv2 SA, an authorized administrator must configure the IPsec VPN client such that algorithms with same strength are used for both IKE Phase 1 and Phase 2 as well as for IKEv2 SA and IKEv2 Child SA.

Windows constructs nonces, which are 32-bit random values, as specified in RFC 2408, Internet Security Association and Key Management Protocol (ISAKMP) section 3.13.[32] When a random number is needed for either a nonce or for key agreement, Windows uses a FIPS-validated random bit generator. When requested, the Windows random bit generator can generate 256 or 512 bits for the caller, the probability of guessing a 256 bit value is 1 in 2^{256} and a 512 bit value is 1 in 2^{512}. When generating the security value *x* used in the IKE Diffie-Hellman key exchange, g^x mod p, Windows uses a FIPS validated random number generator to generate 'x' with length 224, 256, or 384 bits for DH groups 14, 19, and 20 respectively.[33] See the TSS section for **Cryptographic Support** for the NIST CAVP validation numbers.

Windows implements peer authentication using 2048 bit RSA certificates,[34] or ECDSA certificates using the P-256 and P-384 curves for both IKEv1 and IKEv2.[35]

While Windows supports pre-shared IPsec keys, it is not recommended due to the potential use of weak pre-shared keys.[36] Windows simply uses the pre-shared key that was entered by the authorized administrator, there is no additional processing on the input data.

Windows operating systems do not implement the IKEv1 aggressive mode option during a Phase 1 key exchange.

Windows will validate certificates as described in section 6.5.2 by comparing the Common Name of the certificate presented by the VPN gateway to the expected values for the IP address or Fully Qualified Domain Name of the VPN gateway or the user FQDN.

6.3.1.1 RFC Summary

The following table summarizes the use of RFCs and Windows 10:

RFC #	Name	How Used
2407	The Internet IP Security Domain of Interpretation for ISAKMP	Integral part of the Windows Internet Key Exchange (IKE) implementation.
2408	Internet Security Association and Key Management Protocol (ISAKMP)	Integral part of the Windows Internet Key Exchange (IKE) implementation.
2409	The Internet Key Exchange (IKE)	Integral part of the Windows Internet Key Exchange (IKE) implementation.

[31] *Ibid*.
[32] [MS-IKEE], page 51.
[33] http://technet.microsoft.com/en-us/library/cc962035.aspx.
[34] [MS-IKEE], page 73.
[35] http://technet.microsoft.com/en-us/library/905aa96a-4af7-44b0-8e8f-d2b6854a91e6.
[36] http://technet.microsoft.com/en-us/library/cc782582(v=WS.10).aspx.

2986	PKCS #10: Certification Request Syntax Specification; Version 1.7	Public key certification requests issued by Windows.
4106	The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)	Certain IPsec cryptosuites implemented by Windows.
4109	Algorithms for Internet Key Exchange version 1 (IKEv1)	Certain IPsec cryptosuites implemented by Windows.
4301	Security Architecture for the Internet Protocol	Description of the general security architecture for IPsec.
4303	IP Encapsulating Security Payload (ESP)	Specifies the IP Encapsulating Security Payload (ESP) implemented by Windows.
4304	Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)	Specifies a sequence number high-order extension that is implemented by Windows.
4306	Internet Key Exchange (IKEv2) Protocol	Integral part of the Windows Internet Key Exchange (IKE) implementation.
4307	Cryptographic Algorithms for Use in the Internet Key Exchange Version 2 (IKEv2)	Certain IPsec cryptosuites implemented by Windows.
4868	Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec	Certain IPsec cryptosuites implemented by Windows.
4945	The Internet IP Security PKI Profile of IKEv1/ISAKMP, IKEv2, and PKIX	Integral part of the Windows Internet Key Exchange (IKE) implementation.
5280	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	Specifies PKI support implemented by Windows.
5282	Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol	Certain IPsec cryptosuites implemented by Windows.
5996	Internet Key Exchange Protocol Version 2 (IKEv2)	Integral part of the Windows Internet Key Exchange (IKE) implementation.
6379	Suite B Cryptographic Suites for IPsec	Certain IPsec cryptosuites implemented by Windows.

Table 13 IPsec RFCs Implemented by Windows

6.3.2 SFR Mapping

The **Cryptographic Support** function satisfies the following SFRs:

- **FCS_CKM.1(ASYM), FCS_CKM.1(IKE)**: See **Table 11 Cryptographic Standards and Evaluation Methods** and also section 6.2.1 of the Windows 10 [General Purpose] OS security target and section 6.4.1 of the Windows 10 Mobile Device security target for additional verification of key establishment in Windows using CNG.[37]

- **FCS_CKM_EXT.2**: Windows provides secure key storage for private (asymmetric) keys and other data deemed by an authorized subject, such as the pre-shared key, to require secure storage using DPAPI and the NTFS discretionary access control policy.[38]
- **FCS_CKM_EXT.4**: Windows overwrites critical cryptographic parameters immediately after that data is no longer needed.
- **FCS_COP.1(SYM)**: See **Table 11 Cryptographic Standards and Evaluation Methods**.
- **FCS_COP.1(SIGN)**: See **Table 11 Cryptographic Standards and Evaluation Methods**.
- **FCS_COP.1(HASH)**: See **Table 11 Cryptographic Standards and Evaluation Methods**.
- **FCS_COP.1(HMAC)**: See **Table 11 Cryptographic Standards and Evaluation Methods**.
- **FCS_IPSEC_EXT.1**: Windows provides an IPsec implementation as described above in 6.3.1.
- **FCS_RBG_EXT.1**: See **Table 11 Cryptographic Standards and Evaluation Methods**.

6.4 User Data Protection

6.4.1 IPsec VPN Tunnels

The Windows IPsec VPN client can be configured by the device local administrator or the MDM IT administrator, when the device is enrolled. The administrator can also configure the IPsec VPN client that all IP traffic is routed through the IPsec tunnel except for:

- IKE traffic used to establish the VPN tunnel
- IPv4 ARP traffic for resolution of local network layer addresses and to establish a local address
- IPv6 NDP traffic for resolution of local network layer addresses and to establish a local address

The IPsec VPN is an end-to-end internetworking technology and so VPN sessions can be established over physical network protocols such as wireless LAN (Wi-Fi) or local area network.

The components responsible for routing IP traffic through the VPN client:

- The **IPv4 / IPv6 network stack** in the kernel processes ingoing and outgoing network traffic.
- The **IPsec** and **IKE and AuthIP Keying Modules** service which hosts the IKE and Authenticated Internet Protocol (AuthIP) keying modules. These keying modules are used for authentication and key exchange in Internet Protocol security (IPsec).
- The **Remote Access Service** device driver in the kernel, which is used primarily for VPN connections; known as the “RAS IPsec VPN” or “RAS VPN”.
- The **IPsec Policy Agent** service which enforces IPsec policies.

[37] Windows follows section 5 of NIST SP 800-56A.
[38] See https://www.niap-ccevs.org/st/st_vid10677-st.pdf and http://www.commoncriteriaportal.org/files/epfiles/st_windows10.pdf.

6.4.2 Memory Management and Object Reuse

Windows ensures that any previous information content is unavailable upon allocation to subjects and objects. The TSF ensures that resources processed by the kernel or are exported to user-mode processes do not have residual information in the following ways:

- All objects are based on memory and disk storage. Memory allocated for objects, which includes memory allocated for network packets, is either overwritten with all zeros or overwritten with the provided data before being assigned to an object. Read/write pointers prevent reading beyond the space used by the object. Only the exact value of what is most recently written can be read and no more. For varying length objects, subsequent reads only return the exact value that was set, even though the actual allocated size of the object may be greater than this. Objects stored on disk are restricted to only disk space used for that object.
- Subject processes using the IPsec VPN client have associated memory and an execution context. The TSF ensures that the memory associated with subjects is either overwritten with all zeros or overwritten with user data before allocation as described in the previous point for memory allocated to objects. In addition, the execution context (processor registers) is initialized when new threads within a process are created and restored when a thread context switch occurs.
- Network packets processed by IPsec are encrypted in place. In other words, the data to be encrypted is not copied to a separate buffer and then encrypted. The encrypted network packet is encrypted into the same buffer and overwrites the plaintext network packet. The buffers allocated to hold network packets are allocated with enough space to accommodate padding required for encryption. Each network packet is held in its own buffer. There is a list of buffers, one for each packet. A buffer that holds a network packet is not reused for another network packet. After a buffer holding a network packet is no longer in use the memory allocated for the buffer is freed and released back to the TSF.

The above, in combination, will ensure that the memory used for inbound and outbound network packets does not contain data from previous use.

6.4.3 SFR Mapping

The **User Data Protection** function satisfies the following SFR:

- **FDP_IFC_EXT.1**: Windows provides a VPN Client.
- **FDP_RIP.2**: The TSF ensures that previous information contents of resources used for new objects are not discernible in the new object via zeroing or overwriting of memory and tracking read/write pointers for disk storage. Every process is allocated new memory and an execution context. Memory is zeroed or overwritten before allocation.

6.5 Identification and Authentication

6.5.1 IPsec and Pre-shared Keys

IPsec is the only protocol in this evaluation which supports the use of pre-shared keys. These keys can range from a-z, A-Z, the numbers 0 – 9, and any special character entered from the keyboard. The length of the pre-shared key can range from 1 to 100 characters, and so the specific length of 22 characters which the protection profile requires is supported.

The IPsec pre-shared key is used as-is without modification by Windows and so the pre-shared key does not use the Windows random number generator. The reasoning for this is that if the user needs to supply a particular key, that specific key should be used. If the user desires a randomized bit string, then the solution is to use a X.509 certificate which will contain a bit string of suitable length and randomness.

6.5.2 Certificate Validation and Usage

Every Windows component that uses X.509 certificates is responsible for performing certificate validation, however all components use a common subcomponent, which validates certificates as described in RFC 5280 including all applicable usage constraints such as Server Authentication for networking sessions and Code Signing when installing product updates. Every component that uses X.509 certificates will have a repository for public certificates and will select a certificate based on criteria such as entity name for the communication partner, any extended key usage constraints, and cryptographic algorithms associated with the certificate.

If certificate validation fails, or if Windows is not able to check the validation status for a certificate, Windows will not establish a trusted network channel. Certification validation for updates to Windows, mobile applications, and integrity verification is mandatory, neither the administrator nor the user have the option to bypass the results of a failed certificate validation; software installation and updates is further described in **Windows and Application Updates**.

When Windows needs to generate a certificate enrollment request it will include a distinguished name, information about the cryptographic algorithms used for the request, any certification extensions, and information about the client requesting the certificate.

The X.509 certificates are stored in the certificate store for the user or for the machine, depending on whether the IPsec connection is a per-user or a per-machine connection. The physical location of the certificate store is the registry hive for the user or the computer for per-user and per-computer certificates respectively. Certificates can only be loaded into the certificate store by an authorized administrator who has write (i.e., modify) and delete access rights to the registry keys that serve as the certificate store.[39] Certificates can be loaded either using GUI administrator tools, command line tools, or through local and group policy. For a chain of certificates to be validated successfully, as described in the next paragraph, the chain must terminate with a certificate in this Trusted Root Store.

When Windows processes X.509 certificates for IPsec, it follows RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*; which implies that if Windows

[39] Windows makes this determination based on applying the Discretionary Access Control policy which is examined in the Windows General Purpose OS Protection Profile evaluation.

deems that a certificate is invalid, such as for a DN mismatch, a revoked certificate, or an expired certificate, it will not establish an IPsec association. The administrator can use the Set-NetFirewallSetting Cmdlet to manage how IPsec checks certificates. The administrator can create IPsec rules that verify the specific distinguished name in the remote machine's IPsec certificate and fail the connection if the distinguished name is incorrect. The New-NetIPsecAuthProposal Cmdlet with the ValidationCriteria option specifies whether to verify the name in a remote machine's IPsec certificate.

6.5.3 SFR Mapping

The **Identification and Authentication** function satisfies the following SFRs:

- **FIA_PSK_EXT.1**: The TSF allows for the use of pre-shared IPsec keys which are directly used to create an IPsec connection. The set of characters for the pre-shared key is a-z, A-Z, the numbers 0 – 9, and any special character entered from the keyboard.
- **FIA_X509_EXT.1:** Windows follows the certificate revocation procedures specified in RFC 5280.
- **FIA_X509_EXT.1:** Windows used X.509 certificates for, among other uses, IPsec, trusted updates, code-signing and integrity verification.

6.6 Security Management

Windows provides services to identify and authenticate users as described in the Windows 10 General Purpose OS Common Criteria evaluation and the Windows 10 Mobile Device Fundamentals Common Criteria evaluation and so those capabilities are not examined as part of the IPsec VPN Client protection profile evaluation. However, the IPsec VPN Client in Windows does need to manage the X.509 certificates and pre-shared key credentials, described above in **6.5**, which are used to authenticate the IPsec session to the IPsec VPN gateway.

Management Task	Local Administrative Interface	Remote Administrative Interface
Specify VPN gateways to use	• PowerShell • User Interface	• Group Policy • MDM
Specify client credentials to use	• PowerShell • User Interface	• Group Policy • MDM
Configuration of IKE protocol versions	• PowerShell • User Interface	• Group Policy • MDM
Configure IKE authentication techniques	• PowerShell • User Interface	• Group Policy • MDM
Configure the cryptoperiod for the established session keys	• PowerShell	• Group Policy
Configure certificate revocation check	• PowerShell	• Group Policy
Specify the algorithm suites that may be proposed and accepted during the IPsec exchanges	• PowerShell	• Group Policy

Load X.509v3 certificates	• PowerShell • User Interface	• Group Policy • MDM
Update Windows and to verify the updates	• PowerShell • User Interface	• Not included in this evaluation

Table 14 IPsec VPN Client Management Capabilities

In addition, “Configure the cryptoperiod for the established session keys” can be managed by the VPN Gateway.

6.6.1 SFR Mapping

- **FMT_SMF.1(TOE), FMT_SMF.1(MGMT):** Windows provides the authorized administrator with the capability to administer the security functions described in the security target. The mappings to specific functions are described in each applicable section of the TOE Summary Specification.

6.7 Protection of the TSF

The Windows self-tests are a collection of tests which verify that the Windows is operating correctly. The self-tests are enabled when the administrator sets the “System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing” policy; Windows will always run the self-tests described in this section.

The kernel-mode startup self-tests are:[40]

- AES-128 encrypt/decrypt EBC Known Answer Test
- AES-128 encrypt/decrypt CBC Known Answer Test
- AES-128 CMAC Known Answer Test
- AES-128 encrypt/decrypt CCM Known Answer Test
- AES-128 encrypt/decrypt GCM Known Answer Test
- RSA Known Answer Test
- ECDSA sign/verify test on P256 curve
- ECDH secret agreement Known Answer Test on P256 curve
- HMAC-SHA-1 Known Answer Test
- HMAC-SHA-256 and HMAC-SHA-512 Known Answer Tests
- SP800-56A concatenation KDF Known Answer Tests (same as Diffie-Hellman KAT)
- SP800-90 AES-256 counter mode DRBG Known Answer Tests (instantiate, generate and reseed)

The Windows kernel-mode cryptographic module, the Kernel Mode Cryptographic Primitives Library, also performs pair-wise consistency checks upon each invocation of RSA, ECDH, and ECDSA key-pair generation and import as defined in FIPS 140-2. SP 800-56A conditional self-tests are also performed. A continuous RNG test (CRNGT) is used for the random number generators of this cryptographic module. All approved and non-approved RNGs have a CRNGT. The SP 800-90 DRBGs have health tests. A pair-wise consistency test is done for Diffie-Hellman.

[40] When the System Cryptography policy is set, Windows will always perform these self-tests however the evaluated configuration does not use the AES-128 EBC, CMAC, or CCM self-tests.

The Kernel Mode Cryptographic Primitives Library is loaded into the kernel's memory early during the boot process. If there is a failure in any startup self-test, the Kernel Mode Cryptographic Primitives Library DriverEntry function will fail to return the STATUS_SUCCESS status to its caller. The only way to recover from the failure of a startup self-test is to attempt to invoke DriverEntry again, which will rerun the self-tests, and will only succeed if the self-tests passes.

By thoroughly exercising the cryptographic functions, Windows will prevent situations where user data is not transmitted without cryptographic protection.

All operations on the TSF ultimately involve the use of cryptography, and so these tests are sufficient to determine that Windows is operating correctly.

6.7.1 Windows and Application Updates

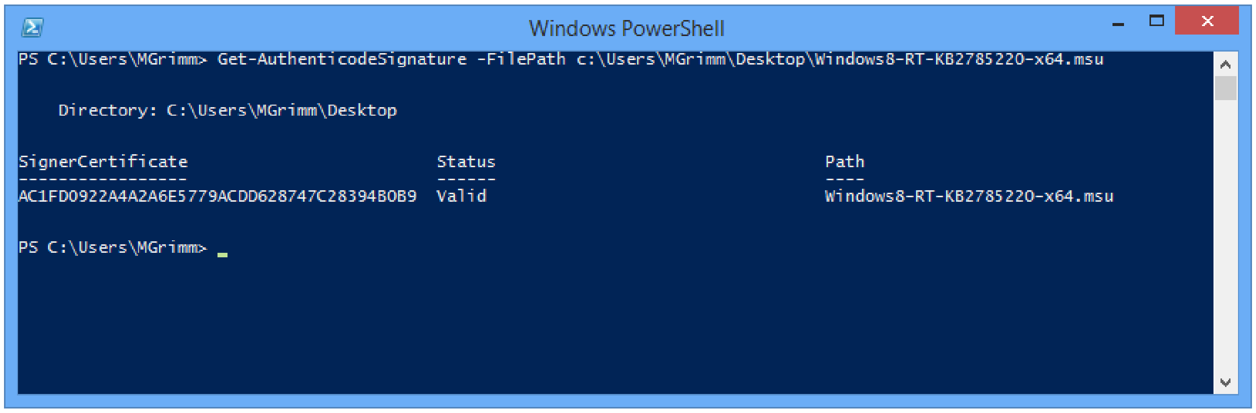
Updates to Windows are delivered as Microsoft Update Standalone Package files (.msu files) and are signed by Microsoft with two digital signatures, a SHA1 signature for legacy applications and a SHA256 signature for modern applications. The RSA SHA256 digital signature is signed by *Microsoft Corporation*, with a certification path through a Microsoft Code Signing certificate and ultimately the Microsoft Root Certification Authority. These certificates are checked by the Windows Trusted Installer prior to installing the update.

The Windows operating system will check that the certificate is valid and has not been revoked using a standard PKI CRL. Once the Trusted Installer determines that the package is valid, it will update Windows; otherwise the installation will abort and there will be an error message in the event log.. Note that the Windows installer will not install an update if the files in the package have lower version numbers than the installed files.

The integrity of the Microsoft Code Signing certificate on the computer is protected by the storage root key within the TPM, and the validated integrity of the Windows binaries as a result of Secure Boot and Code Integrity as described in section 6.6.4, Windows Platform Integrity and Code Integrity, of the Windows 10 General Purpose OS security target.

Updates to Windows are delivered through the Windows Update capability, which is enabled by default, or the user can go to http://www.microsoft.com/security/default.aspx to search and obtain security updates on their own volition.

A user can then check that the signature is valid either by viewing the digital signature details of the file from Windows Explorer or by using the Get-`AuthenticodeSignature` PowerShell Cmdlet. The following is an example of using PowerShell:



```
PS C:\Users\MGrimm> Get-AuthenticodeSignature -FilePath c:\Users\MGrimm\Desktop\Windows8-RT-KB2785220-x64.msu

    Directory: C:\Users\MGrimm\Desktop

SignerCertificate                         Status                                      Path
-----------------                         ------                                      ----
AC1FD0922A4A2A6E5779ACDD628747C28394B0B9  Valid                                       Windows8-RT-KB2785220-x64.msu

PS C:\Users\MGrimm> _
```

If the Get-`AuthenticodeSignature` PowerShell Cmdlet or Windows Explorer could not verify the signature, the status will be marked as invalid. This verification check uses the same functionality described above.

6.7.2 SFR Mapping

The **TSF Protection** function satisfies the following SFRs:

- **FPT_TST_EXT.1**: Windows runs a series of self-tests that confirm that essential cryptographic operations are performed correctly and halts if the self-tests fail. Those cryptographic functions are then used to check integrity of TOE executables as described in **6.7.1**.
- **FPT_TUD_EXT.1**: Windows provides a means to identify the current version of the Windows software. Windows has an update mechanism to deliver updated binaries and a means for a user to confirm that the digital signatures, which ensure the integrity of the update, are valid for both the operating system and Windows Store Applications.

6.8 Trusted Path / Channels

See section **6.3.1**, **IPsec**, for a discussion for how Windows can initiate an IPsec VPN communications channel with a remote VPN gateway that authenticates both ends of the IPsec association and protects data in transit from disclosure and provides detection in case the data was modified in transit. All communication passing through the network connection is protected via IPsec.

6.8.1 SFR Mapping

The **Trusted Path / Channels** function satisfies the following SFRs:

- **FTP_ITC.1**: Windows uses IPsec to provide a trusted communications channel between itself and IPsec VPN peers to provide assured identification of the end point and protects transmitted data from disclosure, detection, and modification.

7 Protection Profile Conformance Claim

This section provides the protection profile conformance claim and supporting justifications and rationale.

7.1 Rationale for Conformance to Protection Profile

This Security Target is in strict compliance with the Protection Profile for IPsec Virtual Private Network (VPN) Clients, version 1.4, October 21, 2013 (IPsec VPN Client PP).

For all of the content incorporated from the protection profile, the corresponding rationale in that protection profile remains applicable to demonstrate the correspondence between the TOE security functional requirements and TOE security objectives.

The requirements in the protection profile are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the functional requirements in the protection profile have been copied into this security target, the dependency analysis for those requirements is not reproduced here.

8 Rationale for Modifications to the Security Requirements

This section provides a rationale that describes how the Security Target reproduced the security functional requirements and security assurance requirements from the protection profile.

8.1 Functional Requirements

This Security Target includes security functional requirements (SFRs) that can be mapped to SFRs found in the protection profile along with SFRs that describe additional features and capabilities. The mapping from protection profile SFRs to security target SFRs along with rationale for operations is presented in Table 15 Rationale for Operations. SFR operations left incomplete in the protection profile have been completed in this security and are identified within each SFR in section 5.1 TOE Security Functional Requirements.

PP Requirement	ST Requirement	Operation & Rationale
FAU_GEN.1	FAU_GEN.1	A selection which is allowed by the PP.
FAU_SEL.1	FAU_SEL.1	Copied from the PP without changes.
FCS_CKM.1(1)	FCS_CKM.1(ASYM)	Three selections which are allowed by the PP.
FCS_CKM.1(2)	FCS_CKM.1(IKE)	Three selections which are allowed by the PP.
FCS_CKM_EXT.2	FCS_CKM_EXT.2	A selection which is allowed by the PP.
FCS_CKM_EXT.4	FCS_CKM_EXT.4	A selection which is allowed by the PP.
FCS_COP.1(1)	FCS_COP.1(SYM)	A selection which is allowed by the PP.
FCS_COP.1(2)	FCS_COP.1(SIGN)	Three selections which are allowed by the PP.
FCS_COP.1(3)	FCS_COP.1(HASH)	Three selections which are allowed by the PP.
FCS_COP.1(4)	FCS_COP.1(HMAC)	Three selections and one assignment which are allowed by the PP.
FCS_IPSEC_EXT.1	FCS_IPSEC_EXT.1	Multiple selections which are allowed by the PP.
FCS_RBG_EXT.1	FCS_RBG_EXT.1	Three selections which are allowed by the PP.
FDP_IFC_EXT.1	FDP_IFC_EXT.1	Copied from the PP without changes.
FDP_RIP.2	FDP_RIP.2	Two selections which are allowed by the PP.
FIA_PSK_EXT.1	FIA_PSK_EXT.1	Three selections which are allowed by the PP.
FIA_X509_EXT.1	FIA_X509_EXT.1	Multiple selections which are allowed by the PP.
FIA_X509_EXT.2	FIA_X509_EXT.2	Multiple selections which are allowed by the PP.

PP Requirement	ST Requirement	Operation & Rationale
FMT_SMF.1	FMT_SMF.1(TOE)	An assignment which is allowed by the PP.
FMT_SMF.1	FMT_SMF.1(MGMT)	A selection which is allowed by the PP.
FPT_TST_EXT.1	FPT_TST_EXT.1	Two selections which are allowed by the PP.
FPT_TUD_EXT.1	FPT_TUD_EXT.1	Four selections which are allowed by the PP.
FTP_ITC.1	FTP_ITC.1	Two selections which are allowed by the PP.

Table 15 Rationale for Operations

8.2 Security Assurance Requirements

The statement of security assurance requirements (SARs) found in section 5.2.2, is in strict conformance with the Protection Profile for IPsec Virtual Private Network (VPN) Clients.

8.3 Rationale for the TOE Summary Specification

This section, in conjunction with section 6, the TOE Summary Specification (TSS), provides evidence that the security functions are suitable to meet the TOE security requirements.

Each subsection in section 6, TOE Security Functions (TSFs), describes a Security Function (SF) of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding SF. The set of security functions work together to satisfy all of the functional requirements. Furthermore, all the security functions are necessary in order for the TSF to provide the required security functionality.

The set of security functions work together to provide all of the security requirements as indicated in Table 16. The security functions described in the TOE Summary Specification and listed in the tables below are all necessary for the required security functionality in the TSF.

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FAU_GEN.1	X								
FAU_SEL.1	X								
FCS_CKM.1(1)		X							
FCS_CKM.1(2)		X							
FCS_CKM_EXT.2		X							

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Resource Utilization	TOE Access	Trusted Path / Channel
FCS_CKM_EXT.4		X							
FCS_COP.1(1)		X							
FCS_COP.1(2)		X							
FCS_COP.1(3)		X							
FCS_COP.1(4)		X							
FCS_IPSEC_EXT.1		X							
FCS_RBG_EXT.1		X							
FDP_IFC_EXT.1			X						
FDP_RIP.2			X						
FIA_PSK_EXT.1				X					
FIA_X509_EXT.1				X					
FIA_X509_EXT.2				X					
FMT_SMF.1(TOE)					X				
FMT_SMF.1(MGMT)					X				
FPT_TST_EXT.1						X			
FPT_TUD_EXT.1						X			
FTP_ITC.1									X

Table 16 Requirement to Security Function Correspondence

9 Appendix A: List of Abbreviations

Abbreviation	Meaning
3DES	Triple DES
ACE	Access Control Entry
ACL	Access Control List
ACP	Access Control Policy
AD	Active Directory
ADAM	Active Directory Application Mode
AES	Advanced Encryption Standard
AGD	Administrator Guidance Document
AH	Authentication Header
ALPC	Advanced Local Process Communication
ANSI	American National Standards Institute
API	Application Programming Interface
APIC	Advanced Programmable Interrupt Controller
BTG	BitLocker To Go
CA	Certificate Authority
CBAC	Claims Basic Access Control, see DYN
CBC	Cipher Block Chaining
CC	Common Criteria
CD-ROM	Compact Disk Read Only Memory
CIFS	Common Internet File System
CIMCPP	Certificate Issuing and Management Components For Basic Robustness Environments Protection Profile, Version 1.0, April 27, 2009
CM	Configuration Management; Control Management
COM	Component Object Model
CP	Content Provider
CPU	Central Processing Unit
CRL	Certificate Revocation List
CryptoAPI	Cryptographic API
CSP	Cryptographic Service Provider
DAC	Discretionary Access Control
DACL	Discretionary Access Control List
DC	Domain Controller
DEP	Data Execution Prevention
DES	Data Encryption Standard
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DFS	Distributed File System
DMA	Direct Memory Access
DNS	Domain Name System
DS	Directory Service
DSA	Digital Signature Algorithm

DYN	Dynamic Access Control
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
EFS	Encrypting File System
ESP	Encapsulating Security Protocol
FEK	File Encryption Key
FIPS	Federal Information Processing Standard
FRS	File Replication Service
FSMO	Flexible Single Master Operation
FTP	File Transfer Protocol
FVE	Full Volume Encryption
GB	Gigabyte
GC	Global Catalog
GHz	Gigahertz
GPC	Group Policy Container
GPO	Group Policy Object
GPOSPP	US Government Protection Profile for General-Purpose Operating System in a Networked Environment
GPT	Group Policy Template
GPT	GUID Partition Table
GUI	Graphical User Interface
GUID	Globally Unique Identifiers
HTTP	Hypertext Transfer Protocol
HTTPS	Secure HTTP
I/O	Input / Output
I&A	Identification and Authentication
IA	Information Assurance
ICF	Internet Connection Firewall
ICMP	Internet Control Message Protocol
ICS	Internet Connection Sharing
ID	Identification
IDE	Integrated Drive Electronics
IETF	Internet Engineering Task Force
IFS	Installable File System
IIS	Internet Information Services
IKE	Internet Key Exchange
IP	Internet Protocol
IPv4	IP Version 4
IPv6	IP Version 6
IPC	Inter-process Communication
IPI	Inter-process Interrupt
IPSec	IP Security
ISAPI	Internet Server API
IT	Information Technology
KDC	Key Distribution Center
LAN	Local Area Network

LDAP	Lightweight Directory Access Protocol
LPC	Local Procedure Call
LSA	Local Security Authority
LSASS	LSA Subsystem Service
LUA	Least-privilege User Account
MAC	Message Authentication Code
MB	Megabyte
MMC	Microsoft Management Console
MSR	Model Specific Register
NAC	(Cisco) Network Admission Control
NAP	Network Access Protection
NAT	Network Address Translation
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
NLB	Network Load Balancing
NMI	Non-maskable Interrupt
NTFS	New Technology File System
NTLM	New Technology LAN Manager
OS	Operating System
PAE	Physical Address Extension
PC/SC	Personal Computer/Smart Card
PIN	Personal Identification Number
PKCS	Public Key Certificate Standard
PKI	Public Key Infrastructure
PP	Protection Profile
RADIUS	Remote Authentication Dial In Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAS	Remote Access Service
RC4	Rivest’s Cipher 4
RID	Relative Identifier
RNG	Random Number Generator
RPC	Remote Procedure Call
RSA	Rivest, Shamir and Adleman
RSASSA	RSA Signature Scheme with Appendix
SA	Security Association
SACL	System Access Control List
SAM	Security Assurance Measure
SAML	Security Assertion Markup Language
SAR	Security Assurance Requirement
SAS	Secure Attention Sequence
SD	Security Descriptor
SHA	Secure Hash Algorithm
SID	Security Identifier
SIP	Session Initiation Protocol
SIPI	Startup IPI

SF	Security Functions
SFP	Security Functional Policy
SFR	Security Functional Requirement
SMB	Server Message Block
SMI	System Management Interrupt
SMTP	Simple Mail Transport Protocol
SP	Service Pack
SPI	Security Parameters Index
SPI	Stateful Packet Inspection
SRM	Security Reference Monitor
SSL	Secure Sockets Layer
SSP	Security Support Providers
SSPI	Security Support Provider Interface
ST	Security Target
SYSVOL	System Volume
TCP	Transmission Control Protocol
TDI	Transport Driver Interface
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module
TSC	TOE Scope of Control
TSF	TOE Security Functions
TSS	TOE Summary Specification
UART	Universal Asynchronous Receiver / Transmitter
UI	User Interface
UID	User Identifier
UNC	Universal Naming Convention
US	United States
UPN	User Principal Name
URL	Uniform Resource Locator
USB	Universal Serial Bus
USN	Update Sequence Number
v5	Version 5
VDS	Virtual Disk Service
VPN	Virtual Private Network
VSS	Volume Shadow Copy Service
WAN	Wide Area Network
WCF	Windows Communications Framework
WebDAV	Web Document Authoring and Versioning
WebSSO	Web Single Sign On
WDM	Windows Driver Model
WIF	Windows Identity Framework
WMI	Windows Management Instrumentation
WSC	Windows Security Center
WU	Windows Update
WSDL	Web Service Description Language

WWW	World-Wide Web
X64	A 64-bit instruction set architecture
X86	A 32-bit instruction set architecture