Microsoft Windows

Common Criteria Evaluation

Microsoft Windows 8

Microsoft Windows RT

Microsoft Windows Server 2012

IPsec VPN Client Security Target

Document Information	
Version Number	1.0
Updated On	January 23 2014

TABLE OF CONTENTS

LIST OF TABLES

1 Security Target Introduction

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
- Describes the organization of the security target

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

ST Title: IPsec VPN Client Security Target

ST Version: version 1.0; January 23, 2014

TOE Software Identification: The IPsec client and dependent functionality from the following Windows Operating Systems (OS) that is needed to comply with the security target:

- Microsoft Windows 8 Edition (32-bit and 64-bit versions)
- Microsoft Windows 8 Pro Edition (32-bit and 64-bit versions)
- Microsoft Windows 8 Enterprise Edition (32-bit and 64-bit versions)
- Microsoft Windows RT
- Microsoft Windows Server 2012 Standard Edition
- Microsoft Windows Server 2012 Datacenter Edition

The following security updates and patches must be applied to the above Windows 8 products:

- All critical security updates published as of February 2013.

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

- All critical security updates published as of February 2013.

The following security updates must be applied to the above Windows Server 2012 products:

- All critical security updates published as of February 2013.

TOE Hardware Identification: The following real and virtualized hardware platforms and components are included in the evaluated configuration:

- Microsoft Surface RT
- Microsoft Surface Pro
- ASUS VivoTab RT
- Dell XPS 10
- Dell OptiPlex 755, 3.0 GHz Intel Core 2 Duo E8400, 64-bit

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

- *Microsoft Windows 8, Microsoft Windows Server 2012, Microsoft Windows RT Common Criteria Supplemental Admin Guidance for IPsec VPN Clients* (version 1.0; May 29, 2013)

Evaluation Assurance: As specified in section 5.2.1 and specific Assurance Activities defined in the protection profile 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, conformant (Part 3 conformant)
- Protection Profile for IPsec Virtual Private Network (VPN) Clients, Version 1.1, December 30, 2012
- CC Part 3 assurance requirements specified in section 5.2.1 and evaluation assurance activities specified in section 5.2.2.

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.
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.
Authenticated Internet Protocol (AuthIP)	The Authenticated Internet Protocol is an extension to the Internet Key Exchange (IKE) Protocol. The Authenticated Internet Protocol supports a more generalized authentication exchange than IKE. This protocol also supports optimizations in key exchange and policy discoverability.[3]
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[4], 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	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,

[1] Hardware and software
[2] Stored or communicated
[3] [MS-AIPS], *Authenticated Internet Protocol,* page 7
[4] According to a defined metric

	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.
Edition	A distinct variation of a Windows OS version. Examples of editions are Windows Server 2012 [Standard] and Windows Server 2012 Datacenter.
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.
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).
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.
Secure State	Condition in which all TOE security policies are enforced.
Security attributes	TOE Security Function (TSF) data associated with subjects, objects and users that is used for the enforcement of the TOE Security Policy (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 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.
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 8, Windows RT and Windows Server 2012 IPsec security target contains the following additional sections:

- TOE Description (Section 2): Provides an overview of the TSF and boundary.
- 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.
- Rationale for the Protection Profile Conformance Claim (Section 7): Presents the rationale concerning compliance of the ST with the IPsec Virtual Private Network (VPN) Client Protection Profile.
- 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 Microsoft Windows 8 operating system, the Microsoft Windows RT operating system, the Microsoft Windows Server 2012 operating system, and supporting hardware.

The focus of this evaluation is on the IPsec Virtual Private Network (VPN) client that is part of the Windows operating system. There are two mechanisms examined in this evaluation that invoke the IPsec VPN client: the Remote Access Service (RAS) VPN interface and the (raw) IPsec interface.

The IPsec interface is a part of the core networking stack and can be used to create IPsec security associations over both local and remote networks.

Remote Access Service (RAS) provides remote access capabilities to client applications on computers running Windows. Historically RAS was used for dial-up and point-to-point networking connections. However it can also be used to establish IPsec virtual private network sessions.

The following sections expand on the networking, cryptographic, authentication, access control, and management aspects of Windows in the context of a protection profile for an IPsec VPN client.

2.1 Product Types

The target of evaluation is the IPsec VPN implementation within Windows 8, Windows RT, and Windows Server 2012.

2.2 Product Description

The TOE is delivered by six product variants of Windows 8, Windows RT, and Windows Server 2012:

- Windows 8
- Windows 8 Pro
- Windows 8 Enterprise
- Windows RT
- Windows Server 2012 Standard
- Windows Server 2012 Datacenter

Windows 8 is suited for business desktops and notebook 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 RT is a new Windows-based operating system that is optimized for thin and light PCs that have extended battery life and are designed for mobile use. Windows RT only runs built-in apps or apps that are downloaded from the Windows Store. Windows RT computers cannot be connected to a Windows domain.

Built for workloads ranging from the department to the enterprise to the cloud, Windows Server 2012 Standard delivers intelligent file and printer sharing, secure connectivity based on Internet technologies, and centralized desktop policy management. It provides the necessary scalable and reliable foundation to support mission-critical solutions for databases, enterprise resource planning software, high-volume, real-time transaction processing, server consolidation, public key infrastructure, virtualization, and additional server roles.

In terms of security and IPsec client functionality, Windows 8, Windows RT, and Server 2012 all share the same security characteristics.

2.3 Security Environment and TOE Boundary

The TOE includes both physical and logical boundaries. Its operational environment is that of a networked environment.

2.3.1 Logical Boundaries

The logical boundary of the TOE includes:

- 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).[5]
- 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.
- The **Cryptographic Services** module which confirms the signatures of Windows program files.
- **Windows Explorer** which can be used to create VPN connections and check the integrity of Windows files and updates.
- The **Certificates** MMC snap-in which is used when the authorized administrator needs to add a certificate, such as a root CA or machine certificate, to their certificate store.[6]
- The **Computer Configuration** MMC snap-in which can be used to set the auditing policy for the computer.
- The **Event Viewer** MMC snap-in which is used to view entries in the audit log.
- The **IP Security Monitor** MMC snap-in which can be used to view active IPsec security associations.
- The **IP Security Policies** MMC snap-in which is used to configure IPsec policies.
- The Windows **Registry** to manually set certain properties for the RAS interface.
- The **netsh** command line application which can be used to manage IPsec settings.
- The **auditpol** command line application which can be used to set the auditing policy for the computer.
- The **sfc** command line application which can be used to check the integrity and repair Windows files.
- The **Get-AuthenticodeSignature PowerShell Cmdlet** which can be used to confirm the signatures of Windows program files.
- The **PowerShell Cmdlets** to manage IPsec:
  - **Get-NetIPsecMainModeSA**
  - **Get-NetIPsecQuickModeSA**
  - **New-NetIPsecAuthProposal**
  - **New-NetIPsecPhase1AuthSet**
  - **New-NetIPsecMainModeCryptoProposal**
  - **New-NetIPsecMainModeCryptoSet**
  - **New-NetIPsecMainModeRule**
  - **New-NetIPsecQuickModeCryptoProposal**

[5] AuthIP key exchange was not examined in the Common Criteria portion of this evaluation.
[6] For common deployment scenarios manually adding these certificates should not be necessary.

- **New-NetIPsecQuickModeCryptoSet**
- **New-NetIPsecRule**
- **Set-NetIPsecMainModeCryptoSet**
- **Set-NetIPsecQuickModeCryptoSet**
- **Set-NetFirewallSetting**

2.3.2 Physical Boundaries

Physically, each TOE tablet, workstation, or server consists of an ARMv7 Thumb-2, x86 or x64 computer. The TOE executes on processors from Intel (x86 and x64), AMD (x86 and x64), Qualcomm (ARM), or NVIDIA (ARM). 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 1.2 or 2.0

While this set of devices is larger than is needed to evaluate IPsec, it is the same set of devices as the General Purpose Operating System Protection Profile evaluation. By using the same set of devices for both evaluations, consumers can gain assurance by using both core OS capabilities and IPsec in combination.

The TOE does not include any network infrastructure components.

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 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.

- **Cryptographic Protection:** Windows provides FIPS-140-2 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,[7] 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 provides object and subject residual information protection.
- **Identification & Authentication**: In the context of this evaluation, Windows provides the ability to use, store, and protect X.509 certificates that are used for IPsec. Windows also has the ability to use pre-shared keys for IPsec.
- **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.
- **Protection of the TOE's 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.

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

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 8, Windows RT, and Windows Server 2012. The assumptions, threats, and policies are copied from the IPsec Client protection profile.

3.1 Threats to Security

Table 3-1 presents known or presumed threats to protected resources that are addressed by Windows 8, Windows RT, and Windows Server 2012 based on conformance to the IPsec VPN Client PP.

Table 3-1 Threats Addressed by Windows 8, Windows RT, and Windows Server 2012

Threat	Description
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 and TOE executable code. 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.UNDETECTED_ACTIONS	Malicious remote users or external IT entities may take actions that adversely affect the security of the TOE. These actions may remain undetected and thus their effects cannot be effectively mitigated.
T.USER_DATA_REUSE	User data may be inadvertently sent to a destination not intended by the original sender.

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 3-2** describes organizational security policies that are addressed by Windows 8, Windows RT, and Windows Server 2012 which are necessary for conformance to the IPsec VPN Client PP.

Table 3-2 Organizational Security Policies

Security Policy	Description
P.COMPATIBILITY	The TOE must meet Request for Comments (RFC) requirements for implemented protocols to facilitate inter-operability with other network equipment using the same protocols.
P.CONFIGURABILITY	The TOE must provide the capability to configure security-relevant aspects of its operation

3.3 Secure Usage Assumptions

Table 3-3 describes the core security assumptions of the environment in which Windows 8, Windows RT, and Windows Server 2012 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 IPsec VPN Client PP:

Table 3-3 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_ADMIN	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

4 Security Objectives

This section defines the security objectives of Windows 8, Windows RT, and Windows Server 2012 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 4-1 describes the security objectives for Windows 8, Windows RT, and Windows Server 2012 which are needed to comply with the IPsec VPN Client PP.

Table 4-1 Security Objectives for the TOE

Security Objective	Source
O.AUTH_COMM	The TOE will provide a means to ensure users are not communicating with some other entity pretending to be the TOE, and that the TOE is communicating with an authorized IT entity and not some other entity pretending to be an authorized IT entity.
O.CRYPTOGRAPHIC_FUNCTIONS	The TOE shall provide cryptographic functions (i.e., encryption/decryption and digital signature operations) to maintain the confidentiality and allow for detection of modification of data that are transmitted outside the TOE and its host environment.
O.GW_AUTHENTICATION	The TOE will authenticate the VPN Gateway that it attempts to establish a security association with.
O.PROTOCOLS	The TOE will ensure that standardized protocols are implemented in the TOE to RFC and/or industry specifications to ensure interoperability.
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.SYSTEM_MONITORING	The TOE will provide the capability to generate audit data.
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

Table 4-2 describes the security objectives for the operational environment as specified in the IPsec VPN Client PP.

Table 4-2 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_ADMIN	TOE Administrators are trusted to follow and apply all administrator guidance in a trusted manner.

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 1.1, December 29, 2011, the Common Criteria, or are defined in the following section.

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.

If general requirements are drawn from the Common Criteria, that is, not from the protection profile, the requirements are copied verbatim, except for some changes to required identifiers to match the iteration convention of this document, and the operations performed in this security target are identified. Note that this security target contains only requirements from the protection profile.

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.

Requirements defined within this security target do not have identified operations.

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 FCS_COP.1(SIGN) (for the component) and FCS_COP.1(SIGN).1 (for the element).
- **Assignment**: Assignments are identified in brackets and bold (e.g., **[assigned value]**).
- **Selection**: Selections are identified in brackets, bold, and italicized (e.g., ***[selected value]***).
    - Assignments within selections are identified using the previous conventions, except that the assigned value would also be italicized and with additional brackets (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.

5.1 TOE Security Functional Requirements

This section specifies the SFRs for the TOE. In some cases, certain capabilities are not available through the RAS interface. Affected SFRs will have a footnote with a clarification.

Table 5-1 TOE Security Functional Requirements

Requirement Class	Requirement Component
Security Audit (FAU)	Audit Data Generation (FAU_GEN.1)
	Selective Audit (FAU_SEL.1)
Cryptographic Support (FCS)	Cryptographic Key Generation for Asymmetric Keys (FCS_CKM.1(ASYM))
	Cryptographic Key Generation for Internet Key Exchange (FCS_CKM.1(IKE))
	Cryptographic Key Zeroization (FCS_CKM_EXT.4)
	Cryptographic operation for Data Encryption/Decryption (FCS_COP.1(SYM))
	Cryptographic operation for Cryptographic Signature (FCS_COP.1(SIGN))
	Cryptographic operation for Cryptographic Hashing (FCS_COP.1(HASH))
	Cryptographic Operation for Keyed-Hash Message Authentication (FCS_COP.1(HMAC))
	Extended: Internet Protocol Security (IPsec) Communications (FCS_IPSEC_EXT)
	Extended: Cryptographic operation for Random Bit Generation (FCS_RBG_EXT.1)
User Data Protection (FDP)	Full Residual Information Protection (FDP_RIP.2)
Identification & Authentication (FIA)	Extended: Pre-Shared Key Composition (FIA_PSK_EXT.1)
	Extended: X.509 Certificates (FIA_X509_EXT.1)
	Extended: X.509 Certificate Storage and Management (FIA_X509_EXT.2)
Security Management (FMT)	Specification of Management Functions (FMT_SMF.1)
Protection of the TSF (FPT)	Extended: TSF Testing (FPT_TST_EXT.1)
	Extended: Trusted Update (FPT_TUD_EXT.1)
Trusted Path/Channels (FTP)	Inter-TSF Trusted Channel (FTP_ITC.1)

5.1.1 Security Audit (FAU)

5.1.1.1 Audit Data Generation (FAU_GEN.1)

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

a) Start-up and shutdown of the audit functions,
b) All auditable events for the not specified level of audit, and
c) All administrative actions,
d) Specifically defined auditable events listed in ~~Table 9~~ **Table 5-2**.

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

a) Date and time of the event, type of event, subject identity, 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 the table below.

Table 5-2 Auditable Events

Requirement	Auditable Events	Additional Audit Record Contents
FAU_GEN.1	None.	
FAU_SEL.1	All modifications to the audit configuration that occur while the audit collection functions are operating.	None.
FCS_CKM.1(ASYM)	Failure of the key generation activity.	None.
FCS_CKM.1(IKE)	Failure of the key generation activity.	None.
FCS_CKM_EXT.4	Failure of the key zeroization process.	Identity of object or entity being cleared.
FCS_COP.1(SYM)	Failure of encryption or decryption.	Cryptographic mode of operation, name/identifier of object being encrypted/decrypted.
FCS_COP.1(SIGN)	Failure of cryptographic signature.	Cryptographic mode of operation, name/identifier of object being signed/verified.
FCS_COP.1(HASH)	Failure of hashing function.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_COP.1(HMAC)	Failure in Cryptographic Hashing for Non-Data Integrity.	Cryptographic mode of operation, name/identifier of object being hashed.
FCS_IPSEC_EXT.1	- Decisions to DISCARD, BYPASS, PROTECT network packets processed by the TOE. - Failure to establish an IPsec SA. - Establishment/Termination of an IPsec SA.	- Presumed identity of source subject. - Identity of destination subject. - Transport layer protocol, if applicable. - Source subject service identifier, if applicable. - The entry in the SPD that applied to the decision. - Reason for failure. - Non-TOE endpoint of connection (IP address) for both successes and failures
FCS_RBG_EXT.1	Failure of the randomization process.	None.
FDP_RIP.2	None.	
FIA_PSK_EXT.1	None.	
FIA_X509_EXT.1	None.	
FIA_X509_EXT.2	None.	
FMT_SMF.1	None.	
FPT_TST_EXT.1	Execution of this set of TSF self-tests. Detected integrity violations.	For integrity violations, the TSF code file that caused the integrity violation.
FPT_TUD_EXT.1	Initiation of the update. Any failure to verify the integrity of the update.	No additional information.

Requirement	Auditable Events	Additional Audit Record Contents
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.

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) [**None**].

5.1.2 Cryptographic Support (FCS)

The functional requirements described in this are only those portions of the cryptographic functions implemented within Windows which are needed to meet the requirements of the IPsec VPN Client protection profile. The intent is to describe only a subset of the product rather than a comprehensive review of Windows.

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 Client protection profile.

FCS_CKM.1(ASYM).1 The TSF 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 [**none**] [8] (as defined in FIPS PUB 186-3, "Digital Signature Standard") [9]
- ***[None]***

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

5.1.2.2 Cryptographic Key Generation for Internet Key Exchange (FCS_CKM.1(IKE))

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

[8] While Windows implements the P-521 curve, it does not expose an IPsec cryptosuite which leverages it.
[9] The RAS interface does not use ECDSA keys.

FCS_CKM.1(IKE).1 The TSF shall generate asymmetric cryptographic keys used for IKE peer authentication in accordance with a:
[
- ***FIPS PUB 186-3, "Digital Signature Standard (DSS)", Appendix B.3 for RSA schemes;***
- ***FIPS PUB 186-3, "Digital Signature Standard (DSS)", Appendix B.4 for ECDSA schemes and implementing "NIST curves" P-256, P-384 and [none];***[10]

]

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

5.1.2.3 Cryptographic Key Zeroization (FCS_CKM_EXT.4)

FCS_CKM_EXT.4.1 The TSF shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

5.1.2.4 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 Client protection profile.

FCS_COP.1.1(SYM) The TSF shall perform encryption and decryption in accordance with a specified cryptographic algorithm AES operating in GCM, CBC [**none**] and cryptographic key sizes 128-bits, 256-bits, and [***192 bits***] that meets the following:

- FIPS PUB 197, "Advanced Encryption Standard (AES)"
- NIST SP 800-38D, NIST SP 800-38A [***NIST SP 800-38C***].

5.1.2.5 Cryptographic Operation for Cryptographic Signature (FCS_COP.1(SIGN))

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

FCS_COP.1.1(SIGN) The TSF shall perform cryptographic signature services in accordance with a:

- RSA Digital Signature Algorithm (rDSA) with a key size (modulus) of 2048 bits or greater that meets FIPS PUB 186-2 or FIPS PUB 186-3, "Digital Signature Standard",
- [***Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater***] that meets FIPS PUB 186-3, "Digital Signature Standard" with "NIST curves" P-256, P-384 and [***none***] (as defined in FIPS PUB 186-3, "Digital Signature Standard (DSA) with a key size (modulus) of 2048 bits or greater that meets FIPS PUB 186-3, "Digital Signature Standard".[11]

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

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

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

FCS_COP.1.1(HASH) The TSF 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 180-3, "Secure Hash Standard."

5.1.2.7 Cryptographic Operation for Keyed-Hash Message Authentication (FCS_COP.1(HMAC))

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

FCS_COP.1.1(HMAC) The TSF 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 180-3, "Secure Hash Standard".

5.1.2.8 Internet Protocol Security (IPsec) Communications (FCS_IPSEC_EXT.1)

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

FCS_IPSEC_EXT.1.2 The TSF shall implement [***transport mode, tunnel mode***].

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

FCS_IPSEC_EXT.1.4 The TSF shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, [***AES-CBC-128, AES-CBC-256 (both specified by RFC 3602) together with a Secure Hash Algorithm (SHA)-based HMAC***].[12]

FCS_IPSEC_EXT.1.5 The TSF 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 TSF 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***].[13]

FCS_IPSEC_EXT.1.7 The TSF shall ensure that IKEv1 Phase 1 exchanges use only main mode.

FCS_IPSEC_EXT.1.8 The TSF shall ensure that [***selection: IKEv2 SA lifetimes can be configured by [an Administrator, VPN Gateway] based on number of packets/number of bytes or 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 or length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs***].[14]

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

[12] The RAS VPN interface does not expose the AES-GCM-128, AES-GCM-256 cryptosuites.
[13] See comment about AES-GCM and the RAS interface.
[14] SA lifetimes for the RAS IPsec VPN security associations are set by the VPN Gateway.

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

FCS_IPSEC_EXT.1.11 The TSF shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), and [***20 (384-bit Random ECP)***].[15]

FCS_IPSEC_EXT.1.12 The TSF shall ensure that all IKE protocols implement peer authentication using a [***RSA, ECDSA***] that use X.509v3 certificates that conform to RFC 4945 and [***Pre-shared Keys***].[16]

FCS_IPSEC_EXT.1.13 The TSF shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 1, IKEv2 IKE_SA***] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 2, IKEv2 CHILD_SA***] connection.

5.1.2.9 Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1 The TSF shall perform all random bit generation (RBG) services in accordance with [***NIST Special Publication 800-90 using*** [***CTR_DRBG (AES), Dual_EC_DRBG (any)***]] seeded by an entropy source that accumulates entropy from [***a combination of hardware-based and software-based noise sources***].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded with a minimum of [***256 bits***] of entropy at least equal to the greatest bit length of the keys and authorization factors that it will generate.

5.1.3 User Data Protection (FDP)

5.1.3.1 Full Residual Data Protection (FDP_RIP.2)

FDP_RIP.2.1 The TSF 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 Pre-Shared Key Composition (FIA_PSK_EXT.1)

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

FIA_PSK_EXT.1.2 The TSF shall be able to accept text-based pre-shared keys that:

- are 22 characters and [**up to 100 characters**];
- composed of any combination of upper and lower case letters, numbers, and special characters (that include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")").

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

[15] The RAS IPsec VPN interface does not offer DH Group 19 as a cryptosuite.
[16] The RAS interface does not use pre-shared keys.

5.1.4.2 X509 Certificates (FIA_X509_EXT.1)

FIA_X509_EXT.1.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec connections.

FIA_X509_EXT.1.3 The TSF shall provide the capability for Administrators to load X.509v3 certificates into the TOE for use by the security functions specified in this PP.

FIA_X509_EXT.1.4 The TSF shall generate a Certificate Request Message as specified in RFC 2986 and be able to provide the following information in the request: public key, Common Name, Organization, Organizational Unit, and Country.

FIA_X509_EXT.1.5 The TSF shall validate the certificate using ***a Certificate Revocation List (CRL) as specified in RFC 5759***].

FIA_X509_EXT.1.6 The TSF shall not establish an SA if a certificate is deemed invalid.

FIA_X509_EXT.1.7 The TSF shall not establish an SA if the distinguished name (DN) contained in a certificate does not match the expected DN for the entity attempting to establish a connection.

FIA_X509_EXT.1.8 When the TSF cannot establish a connection to determine the validity of a certificate, the TSF shall, at the option of the administrator, establish an SA or disallow the establishment of an SA.

5.1.4.3 X.509 Certificate Storage and Management (FIA_X509_EXT.2)

FIA_X509_EXT.1.2 The TSF shall store and protect certificate(s) from unauthorized deletion and modification.

5.1.5 Security Management (FMT)

5.1.5.1 Specification of Management Functions (FMT_SMF.1)

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

- Specify the security associations that shall be proposed and accepted during the IKE negotiations,
- 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,
- Specify authentication methods for peer-to-peer connections, if allowed,
- Ability to update the TOE, and to verify the updates,
- Ability to configure all security management functions identified in other sections of this PP,
- [**None**].

Application Note: These management functions can be invoked locally or remotely through Windows Group Policy.

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 TSF 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 TSF shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the TSF-provided cryptographic services.

5.1.6.2 Extended: Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1 The TSF shall provide authorized administrators the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The TSF shall provide authorized administrators the ability to initiate updates to TOE firmware/software.

FPT_TUD_EXT.1.3 The TSF shall provide a means to verify firmware/software updates to the TOE using a digital signature mechanism and [***published hash***] prior to installing those updates.

5.1.7 Trusted Path/Channels (FTP)

5.1.7.1 Inter-TSF Trusted Channel (FTP_ITC.1)

FTP_ITC.1.1 The TSF 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 TSF shall permit the TSF to initiate communication via the trusted channel.

FTP_ITC.1.3 The TSF shall initiate communication via the trusted channel for all traffic traversing that connection.

5.2 TOE Security Assurance Requirements

The security assurance requirements for the TOE are the requirements defined in the IPsec Client VPN Protection Profile which in turn is based on assurance requirements that are specified in Part 3 of the Common Criteria. No operations are applied to the assurance components.

In addition, the assurance activities from the IPsec Client VPN PP are used to determine that Windows satisfies the IPsec VPN Client security functional requirements. These IPsec VPN Client assurance activities are described in the protection profile and not copied into the security target.

5.2.1 CC Part 3 Assurance Requirements

The following table is the collection of CC Part 3 assurance requirements from the IPsec VPN Client Protection Profile.

Table 5-3 TOE Security Assurance Requirements

Requirement Class	Requirement Component

Development (ADV)	Basic Functional Specification (ADV_FSP.1)
Guidance Documents (AGD)	Operational User Guidance (AGD_OPE.1)
	Preparative Procedures (AGD_PRE.1)
Testing (ATE)	Independent Testing – Conformance (ATE_IND.1)
Vulnerability Assessment (AVA)	Vulnerability Analysis (AVA_VAN.1)
Life-cycle Support (ALC)	Labeling of the TOE (ALC_CMC.1)
	TOE CM Coverage (ALC_CMS.1)

The following sections copy the assurance activities for each family of CC Part 3 assurance requirements.

5.2.1.1 Development

There are no specific assurance activities associated with these SARs. The functional specification documentation is provided to support the evaluation activities described in Section 4.1, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

5.2.1.2 Guidance Documents

5.2.1.2.1 Operational User Guidance

During operation, the activities to be described in the guidance fall into two broad categories; those that are performed by a (non-administrative) user, and those that are performed by an administrator. It should be noted that most procedures needed for non-administrative users are referenced in the assurance activities in Section 4.1.

With respect to the administrative functions, while several have also been described in Section 4.1, additional information is required as follows.

The operational guidance shall at a minimum list the processes running (or that could run) on the TOE in its evaluated configuration during its operation that are capable of processing data received on the network interfaces (there are likely more than one of these, and this is not limited to the process that "listens" on the network interface). It is acceptable to list all processes running (or that could run) on the TOE in its evaluated configuration instead of attempting to determine just those that process the network data. For each process listed, the administrative guidance will contain a short (e.g., one- or two-line) description of the process' function, and the privilege with which the service runs. "Privilege" includes the hardware privilege level (e.g., ring 0, ring 1), any software privileges specifically associated with the process, and the privileges associated with the user role the process runs as or under.

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

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

- For hashes, a description of where the hash for a given update can be obtained. For digital signatures, instructions for obtaining the certificate that will be used by the FCS_COP.1(2) mechanism to ensure that a signed update has been received from the certificate owner. This may be supplied with the product initially, or may be obtained by some other means.
- Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).
- Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature.

5.2.1.2.2 Preparative Procedures

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

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

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

The evaluators shall also perform the following tests:

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

5.2.1.3 Testing

The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the body of this PP's Assurance Activities. While it is not necessary to have one test case per test listed in an Assurance Activity, the evaluators must document in the test plan that each applicable testing requirement in the ST is covered.

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

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

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

5.2.1.4 Vulnerability Assessment

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

5.2.1.5 Lifecycle Support

5.2.1.5.1 Labeling of the TOE

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

5.2.1.5.2 TOE CM Coverage

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

5.2.2 IPsec Client VPN 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

5.2.2.1.1 Audit Data Generation (FAU_GEN.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.[17]

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

[17] This is Table 5-2 in the security target.

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.

This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.[18]

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.

[18] This sentence refers to the assurance activity for FAU_GEN.1.1.

5.2.2.2 Cryptographic Support

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

The evaluator shall use the key pair generation portions of "The FIPS 186-3 Digital Signature Algorithm Validation System (DSA2VS)", "The FIPS 186-3 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS)", and "The RSA Validation System (RSA2VS)" as a guide in testing the requirement above, depending on the selection performed by the ST author. This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

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;

Any TOE-specific extensions, processing that is not included in the documents, or alternative implementations allowed by the documents that may impact the security requirements the TOE is to enforce shall be described.

5.2.2.2.2 Cryptographic Key Generation for Internet Key Exchange (FCS_CKM.1(IKE))

The evaluator shall use the key pair generation portions of "The FIPS 186-3 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS)" and "The RSA Validation System (RSA2VS)" as a guide in testing the requirement above, depending on the selection performed by the ST author. This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

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 FIPS PUB 186-3, the evaluator shall ensure that the TSS contains the following information:

- The TSS shall list all sections of Appendix B 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;

Any TOE-specific extensions, processing that is not included in the Appendices, or alternative implementations allowed by the Appendices that may impact the security requirements the TOE is to enforce shall be described.

5.2.2.2.3 Cryptographic Key Zeroization (FCS_CKM_EXT.4)

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; when they are zeroized (for example, immediately after use, on system shutdown, etc.); and the type of zeroization procedure that is performed (overwrite with zeros, overwrite three times with random pattern, etc.). 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 zeroization procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are zeroized by overwriting once with zeros, while secret keys stored on the internal hard drive are zeroized by overwriting three times with a random pattern that is changed before each write"). If a read-back is done to verify the zeroization, this shall be described as well.

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

The evaluator shall use tests appropriate to the modes selected in the above requirement from "*The Advanced Encryption Standard Algorithm Validation Suite (AESAVS)*", "*The XTS-AES Validation System (XTSVS)*", “*The CMAC Validation System (CMACVS)*", "*The Counter with Cipher Block Chaining-Message Authentication Code (CCM) Validation System (CCMVS)*", and "*The Galois/Counter Mode (GCM) and GMAC Validation System (GCMVS)*" (these documents are available from http://csrc.nist.gov/groups/STM/cavp/index.html) as a guide in testing the requirement above. This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

5.2.2.2.5 Cryptographic Operation for Cryptographic Signature (FCS_COP.1(SIGN))

The evaluator shall use the signature generation and signature verification portions of "*The Elliptic Curve Digital Signature Algorithm Validation System*” (ECDSA2VS), and "*The RSA Validation System*” (RSA2VS) as a guide in testing the requirement above. The Validation System used shall comply with the conformance standard identified in the ST (i.e. FIPS PUB 186-3). This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

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

The evaluator shall use "*The Secure Hash Algorithm Validation System (SHAVS)*" as a guide in testing the requirement above. This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

5.2.2.2.7 Cryptographic Operation for Keyed-Hash Message Authentication (FCS_COP.1(HMAC))

The evaluator shall use "*The Keyed-Hash Message Authentication Code (HMAC) Validation System (HMACVS)*" as a guide in testing the requirement above. This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

5.2.2.2.8 Cryptographic Operation for Random Bit Generation (FCS_RBG_EXT.1)

The evaluator shall review the TSS section to determine the version number of the product containing the RBG(s) used in the TOE. The evaluator shall also confirm that the TSS describes the noise source or sources from which entropy is gathered. The evaluator will further verify that all of the underlying functions and parameters used in the RBG are listed in the TSS.

The evaluator shall verify that the TSS contains a description of the RBG model, including the method for obtaining entropy input, as well as identifying the entropy source(s) used, how entropy is produced/gathered from each source, and how much entropy is produced by each entropy source. The evaluator shall also ensure that the TSS describes the entropy source health tests, a rationale for why the health tests are sufficient to determine the health of the entropy sources, and known modes of entropy source failure. Finally, the evaluator shall ensure that the TSS contains a description of the RBG outputs in terms of the independence of the output and variance with time and/or environmental conditions.

Regardless of the standard to which the RBG is claiming conformance, the evaluator performs the following test:

- Test 1: The evaluator shall determine an entropy estimate for each entropy source by using the Entropy Source Test Suite. The evaluator shall ensure that the TSS includes an entropy estimate that is the minimum of all results obtained from all entropy sources.

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 800-90).

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

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

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

5.2.2.3 IPsec

Note: Because the focus of the evaluation is on IPsec, these assurance activities are presented in a separate section for clarity.

5.2.2.3.1 FCS_IPSEC_EXT.1.1

5.2.2.3.1.1 TSS

The evaluator shall examine the TSS and determine that it describes the rules for processing both inbound and outbound packets in terms of the IPsec policy. As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply (for example, there may be a specific rule that specifies PROTECT, and a general rule that would apply to the same packet that specifies BYPASS). This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

5.2.2.3.1.2 Guidance

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 shall determine that the description in the operational guidance is consistent with the description in the ST, and that the level of detail in the operational guidance is sufficient to allow the administrator to set up the SPD in an unambiguous fashion.

5.2.2.3.1.3 Test

The evaluator uses the operational guidance to configure the TOE 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 several tests that cover a variety of scenarios for packet processing. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the ST. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, for each scenario, that the expected behavior is exhibited, and is consistent with both the ST and the operational guidance.

5.2.2.3.2 FCS_IPSEC_EXT.1.2

5.2.2.3.2.1 TSS

The evaluator checks the TSS to ensure it states that the TOE can operate in tunnel mode and/or transport mode (as selected).

5.2.2.3.2.2 Guidance

The evaluator shall confirm that the operational guidance instructs the Administrator how the TOE is configured in each mode selected.

5.2.2.3.2.3 Test

Test 1 (conditional): If tunnel mode is selected, the evaluator uses the operational guidance to configure the TOE to operate in tunnel mode and also configures a VPN GW to operate in tunnel mode. The evaluator configures the TOE 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 TOE to connect to the VPN GW peer. The evaluator observes 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 to operate in transport mode and also configures a VPN GW to operate in transport mode. The evaluator configures the TOE 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 to connect to the VPN GW. The evaluator observes in the audit trail and the captured packets that a successful connection was established using the transport mode.

5.2.2.3.3 FCS_IPSEC_EXT.1.3

5.2.2.3.3.1 TSS

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.

5.2.2.3.3.2 Guidance

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

5.2.2.3.3.3 Test

Test 1: The evaluator shall configure the TOE's SPD, such that it has entries that contain operations that DISCARD, BYPASS, and PROTECT network packets. The evaluator also configures the TOE so that all auditable events with respect to FCS_IPSEC_EXT.1 are enabled. 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 to the TOE. The evaluator should observe that the network packet is passed by the TOE to the proper destined 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 created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet to the TOE, and observes that the packet was not permitted to flow to any of the TOE's interfaces. The evaluator shall verify that an audit record is generated that specifies that the packet was discarded as expected.

5.2.2.3.4 FCS_IPSEC_EXT.1.4

5.2.2.3.4.1 TSS

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).[19]

5.2.2.3.4.2 Guidance

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

5.2.2.3.4.3 Test

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

[19] This requirement is FCS_COP.1(HMAC) in the security target.

configured to use those algorithms and the evaluator attempts to establish a connection using ESP for those algorithms selected.

5.2.2.3.5 FCS_IPSEC_EXT.1.5

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

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

5.2.2.3.5.3 Test
Test 1: The evaluator shall configure the TOE 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.

5.2.2.3.6 FCS_IPSEC_EXT.1.6

5.2.2.3.6.1 TSS
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.

5.2.2.3.6.2 Guidance
The evaluator ensures that the operational guidance describes how the TOE can be configured to use the mandated algorithms, as well as any additional algorithms selected in the requirement. The guidance is then used to configure the TOE to perform the following test.

5.2.2.3.6.3 Test
Test 1: The evaluator shall configure the TOE to use AES-CBC-128 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 AES-CBC-128. The evaluator will consult the audit trail to confirm the algorithm was that used in the negotiation.

5.2.2.3.7 FCS_IPSEC_EXT.1.7

5.2.2.3.7.1 TSS
The evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol supported by the TOE, 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.

5.2.2.3.7.2 Guidance

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

5.2.2.3.7.3 Test

Test 1 (conditional): The evaluator shall configure the TOE 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.

5.2.2.3.8 FCS_IPSEC_EXT.1.8

5.2.2.3.8.1 TSS

How the lifetimes are established and enforced is described in the RFCs and the evaluator examines the TSS as stated at the beginning of this section.

5.2.2.3.8.2 Guidance

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. 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.

5.2.2.3.8.3 Test

When testing this, 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: 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: 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: 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.

5.2.2.3.9 FCS_IPSEC_EXT.1.9

The evaluator shall check to ensure that, for each DH group supported by the TSF, 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

5.2.2.3.10 FCS_IPSEC_EXT.1.10

The evaluator shall check to ensure that, for each DH group supported by the TSF, 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.

5.2.2.3.11 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 IKE protocols can be successfully completed using that particular DH group.

5.2.2.3.12 FCS_IPSEC_EXT.1.12

5.2.2.3.12.1 TSS

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).[20]

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 for a TOE. The description in the TSS and the operational guidance shall also indicate how pre-shared key establishment is accomplished for both TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

5.2.2.3.12.2 Guidance

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

[20] This requirement is FCS_COP.1(SIGN) in the security target.

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

5.2.2.3.12.3 Test

For efficiency sake, the testing that is performed here has been combined with aspects of the testing for FIA_X509_EXT.1 Extended: X.509 Certificates, specifically FIA_X509_EXT.1.4, and FIA_X509_EXT.1.5.

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 generate a public-private key pair, and submit a CSR (Certificate Signing Request) to a CA (trusted by both the TOE 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 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 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 EP, 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 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 between the TOE and the VPN GW peer. If the

TOE supports generation of the pre-shared key, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE generating the key as well as an instance of the TOE merely taking in and using the key.

5.2.2.3.13 FCS_IPSEC_EXT.1.13

5.2.2.3.13.1 TSS

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.

5.2.2.3.13.2 Guidance

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

5.2.2.3.13.3 Test

Test 1: This test shall be performed for each version of IKE supported by the TOE. 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 by the TOE. 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 by the TOE. 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 by the TOE. 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.

5.2.2.4 User Data Protection

“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.

5.2.2.5 *Identification and Authentication*

5.2.2.5.1 Pre-Shared Key Composition (FIA_PSK_EXT.1)

The evaluator shall examine the operational guidance to determine that it provides guidance to administrators 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 identifies all protocols that allow both text-based and bit-based pre-shared keys, and states that text-based pre-shared keys of 22 characters are supported. If "text-based pre-shared keys" is selected, for each protocol identified by the requirement, 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 the protocol, 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 for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE). Note that one or more of these tests can be performed with a single test case.

- 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.

5.2.2.5.2 X509 Certificates (FIA_X509_EXT.1)

The evaluator shall check the administrative guidance to ensure that it describes how to configure the TOE to use X.509 certificates. This description includes how to load certificates into the TOE, how to generate keys and then make a CRM request to get a certificate for the TOE itself. The description also instructs the administrator how to configure the TOE to either allow or disallow an SA to be established if a decision cannot be made regarding the validity of a certificate being used to authenticate the gateway. In some configurations it may be necessary for the TOE to have established a connection via the VPN Gateway in order to obtain certificate validity information.

The evaluator shall examine the TSS to determine that it describes how the TOE implements certificates to satisfy the requirements. This description includes what aspects are performed by the TOE, and which are allocated to the operational environment.

The testing to ensure the requirements are satisfied is performed in conjunction with the IPsec requirement FCS_IPSEC_EXT.1.12.

5.2.2.5.3 X509 Certificates (FIA_X509_EXT.2)

The evaluator shall ensure the TSS describes all certificate stores implemented that contain certificates used to meet the requirements of this PP. This description shall contain information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access. This description indicates whether the TOE plays any role in the protection of certificates or if it relies solely on the environment to provide the protection.

The evaluator shall examine the guidance documentation to ensure it describes how to configure either the TOE or the environment to prevent unauthorized modification or deletion of the certificates.

The evaluator shall perform the following tests for each function in the system that requires the use of certificates:

- Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function 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.

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place – the TOE, or the environment. It may be that the TOE requests the environment to perform the check and provide a result, or the TOE may do the check itself.

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 environment. 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.

The verification of compliance to these two requirements takes place in the IPsec assurance activities captured in the body of this PP.

5.2.2.6 Security Management

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.

As stated in the application note, a TOE may be configured either locally on the platform, or remotely by a VPN Gateway. The ST will clearly state which functions can be performed locally and remotely. The 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.

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

5.2.2.7 TSF Protection

5.2.2.7.1 Extended: TSF Self-Test (FPT_TST_EXT.1)

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.

The evaluator shall examine the TSS to ensure that it describes how to verify the integrity of stored TSF executable code 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 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. 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.

5.2.2.7.2 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. The evaluator shall perform the following tests:

- 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.

5.2.2.8 Trusted Paths / Channels

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to an access point 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 8, Windows RT, and Windows Server2012 security functions in the context of IPsec and the RAS VPN interface. The Windows 8, Windows RT and Windows Server 2012 Security Functions (SFs) satisfy the security functional requirements of the IPsec VPN Client protection profile.

The TOE performs the following security functions:

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

The following sections present each TOE Security Functions (TSFs) and a mapping of security functions to Security Functional Requirements (SFRs).

6.1 Audit Function

The TOE Audit security function performs:

- Audit Collection
- Selective Audit

6.1.1.1 Audit Collection

The kernel creates the security event log, which contains the security relevant audit records collected on a system. There is one security log (audit log) per machine. A trusted user-mode server process collects audit events from all other parts of the TSF and forwards them to the kernel for storage in the security log. For each audit event, Windows stores the following data in each audit record:

Field in Audit Record	Description
Date	The date the event occurred.
Time	The time the event occurred.
User	The name and security identifier (SID) that represents the user on whose behalf the event occurred.
Event ID	A unique number identifying the particular event class.
Source	The source process or subsystem that generated the event.
Outcome	Indicates whether the 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.

In addition to the system-wide audit policy configuration, it is possible to define a per-user audit policy. This allows individual audit categories (like success or failure) to be enabled or disabled on a per-user basis. 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 identity 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 6-1 Audit Log Event Categories**. These events are also listed in **Table 5-2 Auditable Events.**[21]

Table 6-1 Audit Log Event Categories

Category	Description	FAU_GEN Required Events
System	Audit attempts that affect security of the entire system such as clearing the audit trail.	FAU_SEL.1 FCS_CKM.1* FCS_CKM_EXT.4, FCS_COP.1* FCS_RBG_EXT.1 FMT_SMF.1 FPT_TST_EXT.1 FPT_TUD_EXT.1 FTP_ITC.1
Setup	Integrity violations for Windows updates	FPT_TUD_EXT.1
Logon	Audit attempts to logon or logoff the system, attempts to make a network connection.	FCS_IPSEC_EXT.1

6.1.1.2 Selective Audit

The authorized administrator can select the events to be audited based upon object identity, user identity, computer (host identity), subcategory type (which is a collection of events to audit), and outcome (success or failure) of the event using the **auditpol.exe** command. The *Supplemental Admin Guidance for IPsec VPN Clients* describes the collection of auditing policies which must be enabled to audit the events listed in this security target.

[21] This is a subset of the audit events and audit categories implemented by Windows; it is scoped to the events relevant for the IPsec VPN Client evaluation.

6.1.1.3 Audit Log Overflow Protection

The TSF protects against the loss of events through a combination of controls associated with audit queuing and event logging. Audit data is appended to the audit log until it is full. Windows protects against lost audit data by allowing the authorized administrator to configure the system to generate an audit event when the security log reaches a specified capacity percentage (e.g., 90%). Additionally, the authorized administrator can configure the system to either overwrite the oldest audit record, or not to overwrite events and to shutdown when the security log is full. When so configured, after the system has to shutdown due to audit overflow, only the authorized administrator can log on. When the security log is full, a message is written to the display of the authorized administrator indicating the audit log has overflowed.

This is specified using the **auditpol /set /option:CrashOnAuditFail /value:enable** command.

6.1.2 SFR Mapping:

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

- **FAU_GEN.1**: Windows audit collection is capable of generating audit events for items identified in Table 6-1 Audit Event Categories. 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.
- **FMT_SMF.1**: Only authorized administrators have access to the audit log.
- **FAU_SEL.1**: Windows 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.

6.2 Cryptographic Support

6.2.1 TSS Description

Cryptography API: Next Generation (CNG) API is designed to be extensible at many levels and agnostic to cryptographic algorithm suites and is a part of Windows 8, Windows RT, and Server 2012. An important feature of CNG is its native implementation of the Suite B algorithms. CNG includes support for Suite B that extends to all required algorithms: AES (128, 192, 256 key sizes), 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, mainly used in IPsec), make use of elliptic curve Diffie-Hellman (ECDH) included in Suite B.

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; all are 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.[22, 23]
- A calculated value based on the high-resolution CPU cycle counter .
- Random values gathered periodically from the Trusted Platform Module (TPM), if one is available on the system.
- Random values gathered periodically by calling the RDRAND CPU instruction, if supported by the CPU.

Encryption and decryption operations are performed by independent modules, known as Cryptographic Service Providers (CSPs). The Windows CSPs, specifically the Kernel Cryptographic Primitives Library, are FIPS 140-2 Level 1 compliant. The Kernel Cryptographic Primitives Library is the CSP which is used by the IPsec networking stack.

Windows applies validation techniques to generate symmetric keys in accordance with NIST Special Publication 800-57, "Recommendation for Key Management."

In addition to encryption and decryption services, Windows implements other cryptographic operations such as hashing and digital signatures. The compliance with these cryptographic standards has been demonstrated as follows:

Table 6-2 Cryptographic Standards and Evaluation Methods

Cryptographic Operation	Standard	Evaluation Method
Encryption/Decryption	FIPS 197 AES For ECB, CBC, CFB8, CCM, and GCM modes	NIST CAVP #2197, 2216
Digital signature	FIPS 186-3 DSA	NIST CAVP #687
Digital signature	FIPS 186-3 rDSA	NIST CAVP #1134, #1133
Digital signature	FIPS 186-3 ECDSA	NIST CAVP #341
Hashing	SHA-256, SHA-384, and SHA-512	NIST CAVP #1903
Key Agreement	EC DH	NIST CAVP #36
Keyed-Hash Message Authentication Code	HMAC	NIST CAVP #1345
Random number generation	NIST SP 800-90	NIST CAVP #259 for Dual_EC_DRBG NIST CAVP #258 for CTR_DRBG

The TSF overwrites each intermediate storage area for plaintext key/critical cryptographic security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data). This overwriting is performed by overwriting the memory area with zeros using a forced inline function to minimize execution time. Because the memory word is explicitly set to '0', it is not necessary to do a read-back test.

[22] The Windows OS Loader is the process of being validated as a FIPS 140-2 Level 1 module.
[23] The Windows OS Loader implements a SP 800-90 AES-CTR-DRBG.

The following table describes the secret keys, private keys, and critical security parameters and used by the IPsec VPN Client within Windows:

Table 6-3 IPsec VPN Client Keys and Critical Security Parameters

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 186-3) for IPsec traffic and peer authentication.
Asymmetric ECDSA Private Keys	Keys used for the calculation of ECDSA digital signatures (FIPS 186-3) for IPsec traffic and peer authentication.
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures (FIPS 186-3) for IPsec and signed product updates.
Asymmetric RSA Private Keys	Keys used for the calculation of RSA digital signatures (FIPS 186-3) 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).
DUAL EC DRBG Seed	A secret value maintained internal to the module that provides the seed material for DUAL EC DRBG output (SP 800-90).
DUAL EC DRBG Entropy Input	A secret value maintained internal to the module that provides the entropy material for DUAL EC DRBG output (SP 800-90).
DUAL EC DRBG V	A secret value maintained internal to the module that provides the entropy material for DUAL EC DRBG output (SP 800-90).
DUAL EC DRBG Key	A secret value maintained internal to the module that provides the entropy material for DUAL EC 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.

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

6.2.2 SFR Mapping

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

- **FCS_CKM.1(ASYM), FCS_CKM.1 (IKE)**: See **Table 6-2 Cryptographic Standards and Evaluation Methods**.
- **FCS_CKM_EXT.4**: Windows overwrites critical cryptographic parameters immediately after that data is no longer needed.
- **FCS_COP.1(SYM)**: The TSF uses AES (128-bit and higher key sizes) to encrypt user data and only allows the user who encrypted the data to decrypt the data by ensuring that the SID of the subject requesting decryption is the same as the SID of the subject that requested encryption of the data.
- **FCS_COP.1(SYM), FCS_COP.1 (SIGN), FCS_COP.1 (HASH), FCS_COP.1 (HMAC)**: See **Table 6-2 Cryptographic Standards and Evaluation Methods.**
- **FCS_RBG_EXT.1**: See **Table 6-2 Cryptographic Standards and Evaluation Methods**.

6.3 IPsec

6.3.1 TSS Description

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.[24]

Windows implements both RFCS 2409, Internet Key Exchange (IKEv1), and RFC 4306, Internet Key Exchange version 2, (IKEv2).[25] Windows IPsec supports both tunnel mode and transport mode and provides an option for NAT transversal (reference: section 7.5.5, IPsec Encapsulations).[26] 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.[27] These rules are described in Understanding Firewall Rules and section 5.1 of *Microsoft Windows 8, Microsoft Windows Server 2012, Microsoft Windows RT 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 8, RT, and Server 2012 implement AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256 as encryption algorithms (reference: section 6, Appendix A, Product Behavior).[28] Windows implements HMAC-SHA1 and SHA-256 as authentication algorithms as well as Diffie-Hellman Groups 14,

[24] Also available as [MS-WSO], *Windows System Overview*, page 43 for offline reading.
[25] [MS-IKEE], *Internet Key Exchange Protocol Extensions*, page 8.
[26] [MS-WSO], page 45.
[27] [MS-WPO], page 44.
[28] [MS-IKEE], pages 74 – 75.

19, and 20 (reference: section 6, Appendix A, Product Behavior).[29] This applies to both the encapsulating security payload (ESP) and the encrypted payload in IKEv1 and IKEv2. 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 using a 128 or 256 AES symmetric key to protect the network traffic . 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 same set of cryptosuites are used for both IKE Phase 1 and Phase 2 as well as for IKEv2 SA and IKEv2 Child SA.

Windows constructs nonces as specified in RFC 2408, Internet Security Association and Key Management Protocol (ISAKMP) section 3.13.[30] When a random number is needed for either a nonce or for key agreement, Windows uses a FIPS 140-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 key agreement function.[31] See the TSS section for **Cryptographic Support** for the NIST CAVP validation numbers.

Windows implements peer authentication using 2048 bit RSA certificates,[32] or ECDSA certificates using the P-256 and P-384 curves for both IKEv1 and IKEv2.[33]

While Windows supports pre-shared IPsec keys, it is not recommended due to the potential use of weak pre-shared keys.[34] 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.

6.3.1.1 RFC Summary

The following table summarizes the use of RFCs and Windows 8, Windows RT, and Server 2012:

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.

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

2409	The Internet Key Exchange (IKE)	Integral part of the Windows Internet Key Exchange (IKE) implementation.
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.

6.3.2 TSS Mapping

The IPsec portion of the **Cryptographic Support** function satisfies the following SFRs:

- **FCS_IPSEC_EXT.1**: Windows implements a suite of standards-based networking protocols known collectively as IPsec.
- **FTP_ITC.1**: IPsec provides the trusted channel between a VPN client and a VPN gateway to ensure data confidentiality, integrity and assured delivery to both endpoints.

6.4 User Data Protection

6.4.1 TSS Description

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.

6.4.2 SFR Mapping

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

- **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 TSS Description

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.

Windows generates public key certification requests as described in RFC 2986. A user could request a certificate from a certification authority using the **certreq.exe** command and specifying the Common Name, Organization, Organizational Unit, and Country in the **Subject** key of an .INF file which is included as part of the certification request. By default, **certreq.exe** will create a new public/private key pair, however this can be overridden using the **UseExistingKeySet** key in the request file. Certreq.exe is not intended for normal users and should not be necessary to use for typical IPsec VPN Client scenarios.

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.[35] Certificates can be loaded either using GUI administrator tools, command line tools, or through local and group policy.

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 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 distinguished name in a remote machine's IPsec certificate.

6.5.2 SFR Mapping

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

[35] Windows makes this determination based on applying the Discretionary Access Control policy which is examined as part of the Windows evaluation against the General Purpose Operating System Protection Profile.

- **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:** The TSF also uses X.509 certificates for IPsec connections and follows the certificate revocation procedures specified in RFC 5280.
- **FIA_X509_EXT.2:** The TSF maintains X.509 certificates in the user and machine certificate stores; the Windows operating system limits access to the certificate stores only to authorized administrators.
- **FMT_SMF.1:** The TSF provides the authorized administrator to specify non-default CRL settings.

6.6 Security Management

6.6.1 TSS Description

Using the Network and Sharing Center from the Windows Explorer Control Panel Windows provides the authorized administrator or the VPN Gateway the ability to:

- Specify the security associations that shall be proposed and accepted during the IKE negotiations.
- Configuration of IKE protocol version(s) used.
- Configure IKE authentication techniques used.
- Specify the algorithm suites that may be proposed and accepted during the IPsec exchanges,
- Specify authentication methods for peer-to-peer connections.

These settings can be managed by group policy for domain-connected computers or by local policy for computers that are not domain-joined.[36]

The VPN Gateway is the sole means to configure the cryptoperiod for the established session keys used by the RAS interface. The RAS VPN interface has fixed session lifetimes, but if the VPN Gateway specifies shorter session times, the VPN Gateway session lifetime takes precedence and session keys will be renegotiated. The authorized administrator can configure the cryptoperiod used by IPsec.

The authorized administrator can manage the audit log using the auditpol.exe command line utility or the Computer Management MMC snap-in.

The authorized administrator can verify product updates using the Get-AuthenticodeSignature PowerShell Cmdlet.

6.6.2 SFR Mapping

- **FMT_SMF.1**: 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.

[36] Group policy and local policy security mechanisms will be examined as part of the Windows 8, RT, and Server 2012 OS PP evaluation.

6.7 TSF Protection

6.7.1 TSS Description

6.7.1.1 Windows Self-Tests

The Windows self-tests are a collection of tests which verify that the Windows is operating correctly.

The kernel-mode startup self-tests are:

- 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)
- SP800-90 Dual-EC 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.

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 cryptography that is the basis for IPsec, Windows will avoid situations where data is not transmitted through a non-trusted channel in cases where the authorized administrator has deemed that a trusted channel is necessary.

6.7.1.2 Windows Code Integrity

A Windows operating system verifies the integrity of Windows program code using the Code Integrity capability in Windows. Kernel-mode code signing (KMCS) prevents kernel-mode device drivers, such as the IP v4/v6 stack, from loading unless they are published and digitally signed by developers who have been vetted by one of a handful of trusted certificate authorities (CAs). KMCS, using public-key cryptography technologies, requires that kernel-mode code include a digital signature generated by one of the trusted certificate authorities. When a kernel device driver tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, but using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain that used to verify the signature must match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Windows boot loader.

Starting with the Windows 8 wave of operating systems, code integrity additionally supports larger digital certificates and hash sizes.

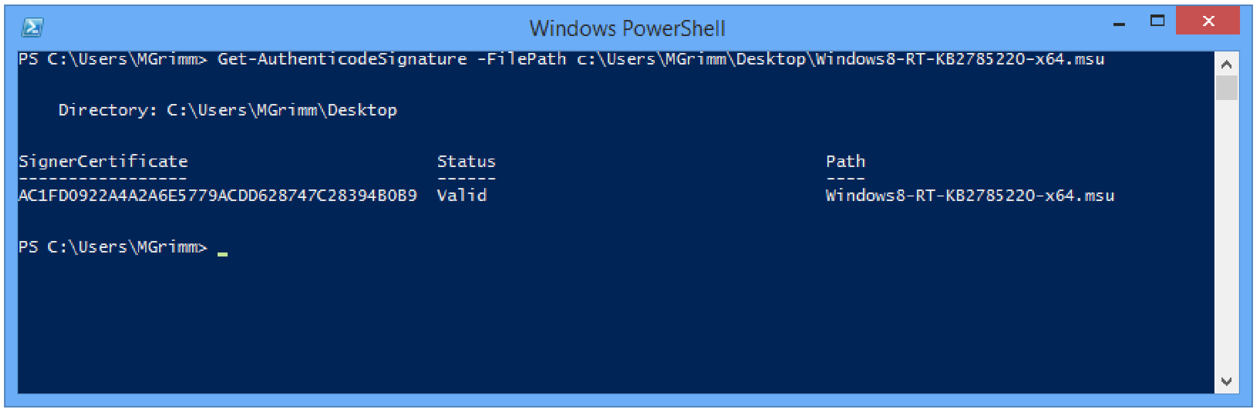
The cryptography used by Code Integrity is validated as part of the Windows FIPS 140 validation.

6.7.1.3 Windows 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.

The Windows operating system will determine that the certificate is valid and has not been revoked using a standard PKI CRL check.

Updates to Windows are delivered through Windows Update, 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 to confirm that essential cryptographic operations are performed correctly and will halt if the self-tests fail. Those cryptographic functions are then used to check integrity of TOE executable files.
- **FPT_TUD_EXT.1**: 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. Windows provides a mechanism to check the current version of the product and binary files that comprise the product.

6.8 Trusted Path / Channels

6.8.1 TSS Description

See **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.

This Common Criteria evaluation examined the Windows mechanisms that can initiate IPsec VPN communications. The first mechanism is a pure IPsec security association with the remote IT entity.

The next mechanism is for a user to create a VPN connection to any VPN Gateway that the user chooses. The security target describes this as the Remote Access Service (RAS) interface. The other mechanism is Direct Access which falls outside the scope of evaluation for the IPsec VPN Client protection profile.

The networking and cryptographic functions are the same for all mechanisms, the difference are the scenarios in which they are used and the available networking options.

6.8.2 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 Rationale for the Protection Profile Conformance Claim

This Security Target is in strict compliance with the Protection Profile for IPsec Virtual Private Network (VPN) Clients, version 1.1, December 30, 2012 ("IPsec VPN Client").

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 for IPsec Virtual Private Network (VPN) Clients are assumed to represent a complete set of requirements that serve to address any interdependencies. Given that all of the functional requirements in the IPsec Client PP 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 Rationale for Modified 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 8-1 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.

Table 8-1 Rationale for Operations

IPsec Client PP Requirement	Security Target Requirement	CC Operation and Rationale
FAU_GEN.1	FAU_GEN.1	A refinement that changes the table identifier.
FAU_SEL.1	FAU_SEL.1	An assignment which is allowed by the PP.
FCS_CKM.1(1)	FCS_CKM.1(AYM)	A selection and assignment which are allowed by the PP.
FCS_CKM.1(2)	FCS_CKM.1(IKE)	A selection which is allowed by the PP.
FCS_CKM_EXT.4	FCS_CKM_EXT.4	Copied from the PP with no operations.
FCS_COP.1(1)	FCS_COP.1(SYM)	An assignment and two selections which are allowed by the PP.
FCS_COP.1(2)	FCS_COP.1(SIGN)	Two selections which are allowed by the PP.
FCS_COP.1(3)	FCS_COP.1(HASH)	Two selections which are allowed by the PP.
FCS_COP.1(4)	FCS_COP.1(HMAC)	An assignment and two selections which are allowed by the PP.
FCS_IPSEC_EXT.1	FCS_IPSEC_EXT.1	Multiple assignments and selections which are allowed by the PP.
FCS_RBG_EXT.1	FCS_RBG_EXT.1	Four selections which are allowed by the PP.
FDP_RIP.2	FDP_RIP.2	A selection which is allowed by the PP.
FIA_PSF_EXT.1	FIA_PSF_EXT.1	An assignment and three

IPsec Client PP Requirement	Security Target Requirement	CC Operation and Rationale
		selections which are allowed by the PP.
FIA_X509_EXT.1	FIA_X509_EXT.1	Copied from the PP with no operations.
FIA_X509_EXT.2	FIA_X509_EXT.2	Copied from the PP with no operations.
FMT_SMF.1	FMT_SMF.1	An assignment which is allowed by the PP.
FPT_TST_EXT.1	FPT_TST_EXT.1	Copied from the PP with no operations.
FPT_TUD_EXT.1	FPT_TUD_EXT.1	A selection which is allowed by the PP.
FTP_ITC.1	FTP_ITC.1	Copied from the PP with no operations.

8.2 Rationale for Modified Assurance Requirements

The statement of security assurance requirements (SARs) found in section 5.2 TOE Security Assurance Requirements, is in strict conformance with the assurance requirements in the IPsec VPN Client protection profile.

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 8-2**. 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.

Table 8-2 Requirement to Security Function Correspondence

Requirement	Audit	Cryptographic Protection	User Data Protection	I & A	Security Management	TSF Protection	Trusted Path / Channel
FAU_GEN.1	X						
FAU_SEL.1	X						
FCS_CKM.1(ASYM)		X					
FCS_CKM.1(IKE)		X					
FCS_CKM_EXT.4		X					
FCS_COA_EXT.1		X					
FCS_COP.1(SYM)		X					
FCS_COP.1(SIGN)		X					
FCS_COP.1(HASH)		X					
FCS_COP.1(HMAC)		X					
FCS_IPSEC_EXT.1		X					
FCS_RBG_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					X		
FPT_TST_EXT.1						X	
FPT_TUD_EXT.1						X	
FTP_ITC.1							X

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
DA	Direct Access
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

10 Appendix B: References

- [MS-AIPS], Authenticated Internet Protocol
- [MS-IKEE], Internet Key Exchange Protocol Extensions
- [MS-WPO], Windows Protocol Overview
- [MS-WSO], Windows System Overview
- Descriptions of the IPsec Algorithms and Methods, Microsoft TechNet article
- Digital Signature Standard (DSS), National Institute of Standards and Technology
- Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, National Institute of Standards and Technology
- Preshared Key Authentication, Microsoft TechNet article
- Secret Key Exchange, Microsoft TechNet

11 Appendix C: Details on Key Establishment and Digital Signatures

This appendix provides additional details about pair-wise key establishment in Windows (FIPS Special Publication 800-56A and FIPS 186-3).

Description Format and Explanation

For each section in SP 800-56A or FIPS 186-3, the descriptions in this appendix use in the following format:

<Section #> <Section Name> -- {“Unimplemented”}[37]

> “Shall not”, “should”, and “should not” Options Implemented by TOE
>
> Rationale for Implementation of “shall not” or “should not”
>
> Omission of Functionality Related to "shall" or “should”

These descriptions have the following meanings:

“Shall not”, “should”, and “should not” Options Implemented by TOE

> 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. A description of “N/A” means it is not applicable because the section does not contain any “shall not”, “should”, or “should not” statements.

Rationale for Implementation of “shall not” or “should not”

> 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. A description of “N/A” means it is not applicable because the TOE does not implement any “shall not” or “should not” functionality, if any.

Omission of Functionality Related to "shall" or “should”

> 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. A description of “N/A” means it is not applicable because the TOE does not omit any “shall” or “should” functionality, if any.

11.1 Special Publication 800-56A

The source document is NIST Special Publication 800-56A, “*Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*”.

[37]The “Unimplemented” tag is optional for those cases in which the TOE does not implement the section.

11.1.1 NIST SP 800-56A Sections

11.1.1.1 Sections 1 – 3

The first three (3) sections do not specify any relevant requirements. For completeness, they are:

1. **Introduction**
2. **Scope and Purpose**
3. **Definitions, Symbols and Abbreviations**

11.1.1.2 Section 4 Key Establishment Schemes Overview

This section is merely a high-level explanation of what key establishment is. Section 4.1 contains the word "shall" and is listed here for completeness.

4.1 Key Agreement Preparations by an Owner

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

4.2 Key Agreement Process
4.3 DLC-based Key Transport Process

11.1.1.3 Section 5 Cryptographic Elements

5.1 Cryptographic Hash Functions

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.2 Message Authentication Code (MAC) Algorithm

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.2.1 MacTag Computation

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.2.2 MacTag Checking

"Shall not", "should", and "should not" Options Implemented by TOE
N/A
Rationale for Implementation of "shall not" or "should not"
N/A
Omission of Functionality Related to "shall" or "should"
N/A

5.2.3 Implementation Validation Message

"Shall not", "should", and "should not" Options Implemented by TOE
N/A
Rationale for Implementation of "shall not" or "should not"
N/A
Omission of Functionality Related to "shall" or "should"
N/A

5.3 Random Number Generation

"Shall not", "should", and "should not" Options Implemented by TOE
N/A
Rationale for Implementation of "shall not" or "should not"
N/A
Omission of Functionality Related to "shall" or "should"
N/A

5.4 Nonces

"Shall not", "should", and "should not" Options Implemented by TOE
The TOE implements random nonces.
Rationale for Implementation of "shall not" or "should not"
N/A
Omission of Functionality Related to "shall" or "should"
N/A

5.5 Domain Parameters

"Shall not", "should", and "should not" Options Implemented by TOE
N/A
Rationale for Implementation of "shall not" or "should not"
N/A
Omission of Functionality Related to "shall" or "should"
N/A

5.5.1 Domain Parameter Generation

This is a section header.

5.5.1.1 FFC Domain Parameter Generation

"Shall not", "should", and "should not" Options Implemented by TOE

The "should" recommendation is:

"If the appropriate security strength does not have an FFC parameter set, then Elliptic Curve Cryptography **should** be used (see Section 5.5.1.2)."

The "should" word only applies to user behavior, which is outside the scope of the TOE.

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.5.1.2 ECC Domain Parameter Generation

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.5.2 Assurances of Domain Parameter Validity

"Shall not", "should", and "should not" Options Implemented by TOE

The "should" recommendation is:

"The application performing the key establishment on behalf of the party **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used."

The "should" word only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.5.3 Domain Parameter Management

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.6 Private and Public Keys

This is a section header with a brief statement as such.

5.6.1 Private/Public Key Pair Generation

This is a section header.

5.6.1.1 FFC Key Pair Generation

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.1.2 ECC Key Pair Generation

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.2 Assurances of the Arithmetic Validity of a Public Key

“Shall not”, “should”, and “should not” Options Implemented by TOE

The “should” recommendation is:

> “The application performing the key establishment on behalf of the owner and recipient **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used.”

The “should” word only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.2.1 Owner Assurances of Static Public Key Validity

“Shall not”, “should”, and “should not” Options Implemented by TOE

The “should” recommendation is:

> “The application performing the key establishment on behalf of the owner **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used.”

The “should” word only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.2.2 Recipient Assurances of Static Public Key Validity

“Shall not”, “should”, and “should not” Options Implemented by TOE

The “should” recommendation is:

> “The application performing the key establishment on behalf of the recipient **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used.”

The “should” word only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.2.3 Recipient Assurances of Ephemeral Public Key Validity

“Shall not”, “should”, and “should not” Options Implemented by TOE

The “should” recommendation is:

“The application performing the key establishment on behalf of the recipient **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used.”

The “should” word only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.2.4 FFC Full Public Key Validation Routine – Unimplemented

Note: Full public key validation is one of several options available for assurances of the arithmetic validity of public keys. Microsoft chose not to implement it in the TOE.

5.6.2.5 ECC Full Public Key Validation Routine – Unimplemented

Note: Full public key validation is one of several options available for assurances of the arithmetic validity of public keys. Microsoft chose not to implement it in the TOE.

5.6.2.6 ECC Partial Public Key Validation Routine

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.3 Assurances of the Possession of a Static Private Key

This section and all subsections concern Owner and Recipient user behavior, which is outside the scope of the TOE.

5.6.3.1 Owner Assurances of Possession of a Static Private Key

5.6.3.2 Recipient Assurance of Owner’s Possession of a Static Private Key

5.6.3.2.1 Recipient Obtains Assurance through a Trusted Third Party

5.6.3.2.2 Recipient Obtains Assurance Directly from the Claimed Owner

5.6.4 Key Pair Management

This is a section header.

5.6.4.1 Common Requirements on Static and Ephemeral Key Pairs

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.4.2 Specific Requirements on Static Key Pairs

“Shall not”, “should”, and “should not” Options Implemented by TOE

The “should” recommendation is:

> “The application performing the key establishment on behalf of the recipient **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used.”

The “should” word only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.6.4.3 Specific Requirements on Ephemeral Key Pairs

“Shall not”, “should”, and “should not” Options Implemented by TOE

1. The first instance of the word “should” is: “An ephemeral key pair should be generated as close to its time of use as possible.” The TOE implements this.
2. The second “should” recommendation is:

   > “The application performing the key establishment on behalf of the recipient **should** determine whether or not to allow key establishment based upon the method(s) of assurance that was used.”

   This second instance of the word “should” only applies to an application, which is outside the scope of the TOE.

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.7 DLC Primitives

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.7.1 Diffie-Hellman Primitives

This is a section header.

5.7.1.1 Finite Field Cryptography Diffie-Hellman (FFC DH) Primitive

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

5.7.1.2 Elliptic Curve Cryptography Cofactor Diffie-Hellman (ECC CDH) Primitive

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.7.2 MQV Primitives -- Unimplemented

This section and all subsections (5.7.2 through 5.7.2.3.2) are MQV primitives. MQV is only one of several options available for key establishment schemes. Microsoft chose not to implement MQV primitives in the TOE.

5.7.2.1 Finite Field Cryptography MQV (FFC MQV) Primitive – Unimplemented
5.7.2.1.1 MQV2 Form of the FFC MQV Primitive – Unimplemented
5.7.2.1.2 MQV1 Form of the FFC MQV Primitive – Unimplemented
5.7.2.2 ECC MQV Associate Value Function – Unimplemented
5.7.2.3 Elliptic Curve Cryptography MQV (ECC MQV) Primitive – Unimplemented
5.7.2.3.1 Full MQV Form of the ECC MQV Primitive – Unimplemented
5.7.2.3.2 One-Pass Form of the ECC MQV Primitive – Unimplemented

5.8 Key Derivation Functions for Key Agreement Schemes

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.8.1 Concatenation Key Derivation Function (Approved Alternative 1)

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

5.8.2 ASN.1 Key Derivation Function (Approved Alternative 2) – Unimplemented

11.1.1.4 Section 6 Key Agreement

This section is an explanation of three (3) categories of key agreement schemes as detailed in sections 6.1, 6.2, and 6.3. Under each category, there are one or more subcategories that are classified by static keys usage. SP 800-56A does not mandate the implementation of all categories and subcategories. Microsoft chose to implement a subset of all possible key agreement schemes in the TOE.

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

The "should" recommendation is:

"Key confirmation may be added to many of these schemes to provide assurance that the participants share the same keying material; see Section 8 for details on key confirmation. Each party should have such assurance."

Microsoft chose not to implement the option of key confirmation in the TOE.

6.1 Schemes Using Two Ephemeral Key Pairs, C(2)

This section is a header with a short explanation of the subcategories.

6.1.1 Each Party Has a Static Key Pair and Generates an Ephemeral Key Pair, C(2, 2) – Unimplemented

This section and all subsections (6.1.1 through 6.1.1.5) are optional. Microsoft chose not to implement them in the TOE.

6.1.1.1 dhHybrid1, C(2, 2, FFC DH) – Unimplemented
6.1.1.2 Full Unified Model, C(2, 2, ECC CDH) – Unimplemented
6.1.1.3 MQV2, C(2, 2, FFC MQV) – Unimplemented
6.1.1.4 Full MQV, C(2, 2, ECC MQV) – Unimplemented
6.1.1.5 Rationale for Choosing a C(2, 2) Scheme – Unimplemented

6.1.2 Each Party Generates an Ephemeral Key Pair; No Static Keys are Used, C(2, 0)

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

6.1.2.1 dhEphem, C(2, 0, FFC DH)

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

6.1.2.2 Ephemeral Unified Model, C(2, 0, ECC CDH)

"Shall not", "should", and "should not" Options Implemented by TOE

N/A

Rationale for Implementation of "shall not" or "should not"

N/A

Omission of Functionality Related to "shall" or "should"

N/A

6.1.2.3 Rationale for Choosing a C(2, 0) Scheme

This section only explains the rationale.

6.2 Schemes Using One Ephemeral Key Pair, C(1)

This section is a header with a short explanation of the subcategories.

6.2.1 Initiator Has a Static Key Pair and Generates an Ephemeral Key Pair; Responder Has a Static Key Pair, C(1, 2) – Unimplemented

This section and all subsections (6.2.1 through 6.2.1.5) are optional. Microsoft chose not to implement them in the TOE.

6.2.1.1 dhHybridOneFlow, C(1, 2, FFC DH) – Unimplemented
6.2.1.2 One-Pass Unified Model, C(1, 2, ECC CDH) – Unimplemented
6.2.1.3 MQV1, C(1, 2, FFC MQV) – Unimplemented
6.2.1.4 One-Pass MQV, C(1, 2, ECC MQV) – Unimplemented
6.2.1.5 Rationale for Choosing a C(1, 2) Scheme – Unimplemented

6.2.2 Initiator Generates Only an Ephemeral Key Pair; Responder Has Only a Static Key Pair, C(1, 1)

“Shall not”, “should”, and “should not” Options Implemented by TOE
N/A
Rationale for Implementation of “shall not” or “should not”
N/A
Omission of Functionality Related to "shall" or “should”
N/A

6.2.2.1 dhOneFlow, C(1, 1, FFC DH)

“Shall not”, “should”, and “should not” Options Implemented by TOE
N/A
Rationale for Implementation of “shall not” or “should not”
N/A
Omission of Functionality Related to "shall" or “should”
N/A

6.2.2.2 One-Pass Diffie-Hellman, C(1, 1, ECC CDH)

“Shall not”, “should”, and “should not” Options Implemented by TOE
N/A
Rationale for Implementation of “shall not” or “should not”
N/A
Omission of Functionality Related to "shall" or “should”
N/A

6.2.2.3 Rationale in Choosing a C(1, 1) Scheme

This section only explains the rationale.

6.3 Scheme Using No Ephemeral Key Pairs, C(0, 2)

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

6.3.1 dhStatic, C(0, 2, FFC DH)

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

6.3.2 Static Unified Model, C(0, 2, ECC CDH)

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

6.3.3 Rationale in Choosing a C(0, 2) Scheme

This section only explains the rationale.

11.1.1.5 Section 7 DLC-Based Key Transport

This section was not selected in the ST.

11.1.1.6 Section 8 Key Confirmation

As allowed in Section 6 Key Agreement, Microsoft chose not to implement optional key confirmation in the TOE.

11.1.1.7 Section 9 Key Recovery

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

11.1.1.8 Section 10 Implementation Validation

The TOE shall be proven to comply with the “shall” statements in this section as evidenced by NIST CMVP FIPS 140-2 validation certificates when they are published on the NIST CMVP Validated FIPS 140-1 and FIPS 140-2 Cryptographic Modules website:

http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm

“Shall not”, “should”, and “should not” Options Implemented by TOE
N/A
Rationale for Implementation of “shall not” or “should not”
N/A
Omission of Functionality Related to "shall" or “should”
N/A

11.1.1.9 Appendices A, D, and E (Informative)

These appendices are informative and are included here for completeness.

11.1.1.10 Appendix B: Rationale for Including Identifiers in the KDF Input

This section is explanatory rationale and is included here for completeness.

11.1.1.11 Appendix C: Data Conversions (Normative)

“Shall not”, “should”, and “should not” Options Implemented by TOE
N/A
Rationale for Implementation of “shall not” or “should not”
N/A
Omission of Functionality Related to "shall" or “should”
N/A

11.1.2 Exceptions

11.1.2.1 TOE-Specific Extensions

There are not any TOE-specific extensions that may impact the security requirements the TOE is to enforce.

11.1.2.2 Additional Processing

There is no processing that is not included in the documents that may impact the security requirements the TOE is to enforce.

11.1.2.3 Alternative Implementations

There are no alternative implementations allowed by the documents that may impact the security requirements the TOE is to enforce.

11.2 FIPS 186-3

The source document is FIPS 186-3, “*Digital Signature Standard (DSS)*”.

11.2.1 Sections from FIPS 186-3 Appendix B Key Pair Generation

This part covers the compliant sections from FIPS PUB 186-3, “Digital Signature Standard (DSS)”, Appendix B. The introduction of Appendix B includes a “shall” statement for bit string to integer conversion, which the TOE implements.

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

11.2.1.1 B.1 FFC KEY PAIR GENERATION

This section has two subsections (B.1.1 and B.1.2) of which only one must be implemented. Microsoft chose to implement only B.1.2.

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

B.1.1 Key Pair Generation Using Extra Random Bits -- Unimplemented

B.1.2 Key Pair Generation by Testing Candidates

“Shall not”, “should”, and “should not” Options Implemented by TOE

None Implemented

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

The “should” recommendation is:

> “If an error is encountered during the generation process, invalid values for *x* and *y* **should** be returned, as represented by *Invalid_x* and *Invalid_y* in the following specification.”

The TOE does not implement the return of invalid values if an error is encountered. A function generates a random number of the correct length and tries again if the number is not in the correct range.

11.2.1.2 B.2 FFC PER-MESSAGE SECRET NUMBER GENERATION

This section has two subsections (B.2.1 and B.2.2) of which only one must be implemented. Microsoft chose to implement only B.2.2.

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

B.2.1 Per-Message Secret Number Generation Using Extra Random Bits – Unimplemented

B.2.2 Per-Message Secret Number Generation by Testing Candidates

“Shall not”, “should”, and “should not” Options Implemented by TOE

None Implemented

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

The “should” recommendation is:

“If an error is encountered during the generation process, invalid values for *k* and k^{-1} **should** be returned, as represented by *Invalid_k* and *Invalid_k_inverse* in the following specification.”

The TOE does not implement the return of invalid values if an error is encountered. A function generates a random number of the correct length and tries again if the number is not in the correct range.

11.2.1.3 B.3 IFC KEY PAIR GENERATION

This is a section header.

B.3.1 Criteria for IFC Key Pairs

Note: Section B.3.6 is implemented in the TOE and meets the “shall” requirement in Section B.3.1, method B, case #3. As permitted by the standard, no other B.3 subsections are implemented.

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

B.3.2 Generation of Random Primes that are Provably Prime – Unimplemented
B.3.2.1 Get the Seed – Unimplemented
B.3.2.2 Construction of the Provable Primes p and q – Unimplemented
B.3.3 Generation of Random Primes that are Probably Prime – Unimplemented

Note: B.3.3 code exists, but it is not called, therefore, it is labeled as “Unimplemented”.

B.3.4 Generation of Provable Primes with Conditions Based on Auxiliary Provable Primes – Unimplemented
B.3.5 Generation of Probable Primes with Conditions Based on Auxiliary Provable Primes – Unimplemented
B.3.6 Generation of Probable Primes with Conditions Based on Auxiliary Probable Primes

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

11.2.1.4 B.4 ECC KEY PAIR GENERATION

“Shall not”, “should”, and “should not” Options Implemented by TOE

N/A

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

N/A

B.4.1 Key Pair Generation Using Extra Random Bits – Unimplemented

B.4.2 Key Pair Generation by Testing Candidates

“Shall not”, “should”, and “should not” Options Implemented by TOE

None Implemented

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

The “should” recommendation is:

“If an error is encountered during the generation process, invalid values for *d* and *Q* **should** be returned, as represented by *Invalid_d* and *Invalid_Q* in the following specification.”

The TOE does not implement the return of invalid values if an error is encountered. A function generates a random number of the correct length and tries again if the number is not in the correct range.

11.2.1.5 B.5 ECC PER-MESSAGE SECRET NUMBER GENERATION

This section has two subsections (B.5.1 and B.5.2) of which only one must be implemented. Microsoft chose to implement only B.5.2.

B.5.1 Per-Message Secret Number Generation Using Extra Random Bits – Unimplemented

B.5.2 Per-Message Secret Number Generation by Testing Candidates

“Shall not”, “should”, and “should not” Options Implemented by TOE

None Implemented

Rationale for Implementation of “shall not” or “should not”

N/A

Omission of Functionality Related to "shall" or “should”

The “should” recommendation is:

“If an error is encountered during the generation process, invalid values for *k* and k^{-1} **should** be returned, as represented by *Invalid_k* and *Invalid_k_inverse* in the following specification.”

The TOE does not implement the return of invalid values if an error is encountered. A function generates a random number of the correct length and tries again if the number is not in the correct range.

11.2.2 Exceptions

11.2.2.1 TOE-Specific Extensions
There are not any TOE-specific extensions that may impact the security requirements the TOE is to enforce.

11.2.2.2 Additional Processing
There is no processing that is not included in the documents that may impact the security requirements the TOE is to enforce.

11.2.2.3 Alternative Implementations
There are no alternative implementations allowed by the documents that may impact the security requirements the TOE is to enforce.