Assurance Activities Report

for

Aruba Virtual Intranet Access (VIA) Client Version 4.3

Version 1.1

August 24, 2022

Evaluated By:

Leidos Inc.
https://www.leidos.com/civil/commercial-cyber/product-compliance
Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, MD 21046

Prepared for:

National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

The Developer of the TOE:

Aruba, a Hewlett Packard Enterprise Company
3333 Scott Blvd
Santa Clara, CA 94089

The TOE Evaluation was Sponsored by:

Aruba, a Hewlett Packard Enterprise Company
3333 Scott Blvd
Santa Clara, CA 94089

Evaluation Personnel:

Justin Fisher
Allen Sant

Revision History

Version	Date	Description
1.0	August 8, 2022	Initial version
1.0	August 24, 2022	Updates based on reviewer feedback

Contents

1 Introduction

This document presents the results of performing assurance activities associated with the Aruba (a Hewlett Packard Enterprise company) Virtual Intranet Access (VIA) Client Version 4.3 evaluation. This report contains sections documenting the performance of assurance activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in the following Protection Profile and PP-Module:

- Protection Profile for Application Software, Version 1.3, 01 March 2019 [APP PP]
- PP-Module for VPN Clients, Version 2.3, 10 August 2021 [MOD VPNC]

Note that, in accordance with NIAP Policy Letter #5, all cryptography in the TOE for which NIST provides validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components must be NIST validated. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity. As such, Test assurance activities associated with functional requirements within the scope of Policy Letter #5 are performed by verification of the relevant CAVP certification and not through performance of any testing as specified in the PP or its supporting document.

This evaluation claimed the following optional, selection-based, and objective SFRs:

[APP PP]: FCS_CKM.1(2) (optional), FCS_RBG_EXT.2 (selection-based), FCS_CKM.1(1) selection-based), FCS_CKM.2 (selection-based), FCS_COP.1(1) (selection-based) FCS_COP.1(2) (selection-based), FCS_COP.1(3) (selection-based), FCS_COP.1(4) (selection-based), FIA_X509_EXT.1 (selection-based), FIA_X509_EXT.2 (selection-based), FPT_TUD_EXT.2 (selection-based)

[MOD VPNC]: FIA_PSK_EXT.1 (selection-based)

1.1 Technical Decisions

The NIAP Technical Decisions referenced below apply to [APP PP] and [MOD VPNC]. Rationale is included for those Technical Decisions that do not apply to this evaluation.

TD0416 – Correction to FCS_RBG_EXT.1 Test Activity

This TD applies to the TOE and has been incorporated into the AAR.

TD0427 – Reliable Time Source

This TD applies to the TOE but only modifies an assumption.

TD0434 – Windows Desktop Applications Test

This TD applies to the TOE and has been incorporated into the AAR.

TD0435 – Alternative to SELinux for FPT_AEX_EXT.1.3

This TD applies to the TOE and has been incorporated into the AAR.

TD0437 – Supported Configuration Mechanism

This TD applies to the TOE and has been incorporated into the AAR.

TD0445 – User Modifiable File Definition

This TD applies to the TOE and has been incorporated into the AAR.

TD0465 – Configuration Storage for .NET Apps

This TD applies to the TOE and has been incorporated into the AAR.

TD0495 – FIA_X509_EXT.1.2 Test Clarification

This TD applies to the TOE and has been incorporated into the AAR.

TD0498 – Application Software PP Security Objectives and Requirements Rationale

This TD applies to the TOE but only modifies the Security Objectives Rationale and adds the SFR Rationale.

TD0510 – Obtaining random bytes for iOS/macOS

This TD does not apply to the TOE because the TOE does not have an iOS or macOS platform version.

TD0515 – Use Android APK manifest in test

This TD applies to the TOE and has been incorporated into the AAR.

TD0519 – Linux symbolic links and FMT_CFG_EXT.1

This TD applies to the TOE and has been incorporated into the AAR.

TD0543 – FMT_MEC_EXT.1 evaluation activity update

This TD applies to the TOE and has been incorporated into the AAR.

TD0544 – Alternative testing methods for FPT_AEX_EXT.1.1

This TD applies to the TOE and has been incorporated into the AAR.

TD0548 – Integrity for installation tests in AppSW PP 1.3

This TD applies to the TOE and has been incorporated into the AAR.

TD0554 – iOS/iPadOS/Android AppSW Virus Scan

This TD applies to the TOE and has been incorporated into the AAR.

TD0561 – Signature verification update

This TD applies to the TOE and has been incorporated into the AAR.

TD0582 – PP-Configuration for Application Software and Virtual Private Network (VPN) Clients now allowed

This TD applies to the TOE; the TD itself does not change anything in the AAR.

TD0598 – Expanded AES Modes in FCS_COP for App PP

This TD is applicable to the TOE but does not affect it as the TD adds selectable options for FCS_COP.1(1) that the ST does not choose.

TD0600 – Conformance claim sections updated to allow for MOD_VPNC_V2.3

This TD applies to the TOE; the TD itself does not change anything in the AAR.

TD0601 – X.509 SFR Applicability in App PP

This TD applies to the TOE and has been incorporated into the AAR.

TD0622 – VPNC MOD FTP_DIT_EXT.1 corrections

This TD applies to the TOE; the only change resulting from the TD is SFR text so it does not affect the AAR.

1.2 References

[ST]	Aruba Virtual Intranet Access (VIA) Client Version 4.3 Security Target, Version 1.0, August 23, 2022
[APP PP]	Protection Profile for Application Software, Version 1.3, 01 March 2019
[MOD VPNC]	PP-Module for VPN Clients, Version 2.3, 10 August 2021
[CCECG]	Aruba Virtual Intranet Access (VIA) 4.x Client Common Criteria Guidance, Version 1.2, March 2022
[User]	Aruba Virtual Intranet Access (VIA) 4.x Help Center, https://www.arubanetworks.com/techdocs/VIA/4x/Content/home.htm, retrieved August 23, 2022
[Test]	Aruba VIA 4.3 Common Criteria Test Report and Procedures, Version 1.2, 24 August 2022
[VA]	Aruba Virtual Intranet Access (VIA) Client Version 4.3- Vulnerability Analysis, Version 1.3, 24 August 2022

1.3 Conformance Claims

Common Criteria Version:

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

Common Evaluation Methodology Version:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017.

Protection Profiles:

- PP-Configuration for Application Software and Virtual Private Network Clients, Version 1.0, 13 August 2021 (exact conformance), which includes the following components:
  - Protection Profile for Application Software, Version 1.3, 01 March 2019
  - PP-Module for VPN Clients, Version 2.3, 10 August 2021

1.4 SAR Evaluation

The following Security Assurance Requirements (SARs) were evaluated during the evaluation of the TOE:

SAR	Verdict
ASE_CCL.1	Pass
ASE_ECD.1	Pass
ASE_INT.1	Pass
ASE_OBJ.2	Pass
ASE_REQ.2	Pass
ASE_TSS.1	Pass
ADV_FSP.1	Pass
AGD_OPE.1	Pass
AGD_PRE.1	Pass
ALC_CMC.1	Pass
ALC_CMS.1	Pass
ALC_TSU_EXT.1	Pass
ATE_IND.1	Pass
AVA_VAN.1	Pass

The evaluation work units are listed in the proprietary ETR, except for ALC_TSU_EXT.1, which is an extended SAR defined in [App PP] and is evaluated solely through the completion of the evaluation activities in this AAR. The evaluators note per the PP evaluation activities that many of the SARs were successfully evaluated through completion of the associated evaluation activities present in the claimed PP.

2 Security Functional Requirement Assurance Activities

This section describes the assurance activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The assurance activities are derived from both [APP PP] and [MOD VPNC]. As such, this report identifies the source of each SFR (and assurance activity if appropriate). NIAP Technical Decisions have been applied and are identified as appropriate.

2.1 Cryptographic Support (FCS)

The following table is reproduced from [ST] and identifies all claimed SFRs that are satisfied by the issuance of NIST CAVP certificates along with the associated certificate, in accordance with NIAP Policy #5. The same combined algorithm validation (A2147) included operational environments for all three tested platform versions (Windows, Android, Linux). The hardware platforms on which the TOE was tested align with the certified operational environments per the certificate worksheet in the proprietary Evaluation Technical Report.

Requirements	Functions	Certificate		
		Windows	Android	Linux
Cryptographic Asymmetric Key Generation				
FCS_CKM.1(1)	ECC schemes using “NIST curves” P-256, P-384 that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4	A2147	A2147	A2147
	FFC schemes] using Diffie-Hellman group 14 that meet the following: RFC 3526, Section 3	A2147	A2147	A2147
Cryptographic Key Generation (IKE)				
FCS_CKM.1.1/VPN	FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.3 for RSA schemes;	A2147	A2147	A2147
	FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4 for ECDSA schemes and implementing “NIST curves”, P-256 and P-384	A2147	A2147	A2147
Cryptographic Key Establishment				
FCS_CKM.2.1	Elliptic curve-based key establishment schemes] that meets the following: NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”	A2147	A2147	A2147
	Key establishment scheme using Diffie-Hellman group 14 that meets the following: RFC 3526, Section 3	A2147	A2147	A2147
Cryptographic Operation – Encryption/Decryption				
FCS_COP.1.1(1)	AES-CBC (128-bit, 256-bit) as defined in NIST SP 800-38A	A2147	A2147	A2147

Requirements	Functions	Certificate		
		Windows	Android	Linux
	AES-GCM (128-bit, 256-bit) as defined in NIST SP 800-38D			
Cryptographic Operation - Hashing				
FCS_COP.1.1(2)	SHA-1, SHA-256, and SHA-384 as FIPS Pub 180-4	A2147	A2147	A2147
Cryptographic Operation - Signing				
FCS_COP.1.1(3)	RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 4	A2147	A2147	A2147
	ECDSA schemes using "NIST curves" P-256, P-384 that meet FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5	A2147	A2147	A2147
Cryptographic Operation - Keyed-Hash Message Authentication				
FCS_COP.1.1(4)	HMAC-SHA-1 HMAC-SHA-256 HMAC-SHA-384 That meets the following: FIPS Pub 198-1 and FIPS Pub 180-4	A2147	A2147	A2147
Random Bit Generation Services Random Bit Generation from Application				
FCS_RBG_EXT.1 FCS_RBG_EXT.2	256-bits NIST Special Publication 800-90A using CTR_DRBG (AES)	A2147	A2147	A2147

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

2.1.1.1 TSS Activities

The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

[ST] section 6.1.1 indicates that the TOE supports ECC using P-256 and P-384 for IPsec DH groups 19 and 20, and FFC using 2048-bit keys for IPsec DH group 14.

If the application **invokes platform-provided functionality for asymmetric key generation**, then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

"Invoke platform-provided functionality" is not claimed so this is not applicable to the TOE.

2.1.1.2 Guidance Activities

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

[CCECG] notes in section 1 that the TOE's configuration settings are acquired from the Mobility Controller. The configuration references in section 2.2.4 and 2.2.5 of [CCECG] summarizes how to configure the connection settings for IKEv1 and IKEv2 policies, respectively. In both of these sections, information is present on how to select the Diffie-Hellman group that is used for ISAKMP. The key generation and key establishment scheme used by the TOE is implicitly configured through specifying the Diffie-Hellman group that is used.

2.1.1.3 Test Activities

If the application implements asymmetric key generation, then the following test activities shall be carried out.

Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application.

Key Generation for FIPS PUB 186-4 RSA Schemes

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

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

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

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- $n = p \cdot q$,

- p and q are probably prime according to Miller-Rabin tests,
- GCD(p-1,e) = 1,
- GCD(q-1,e) = 1,
- 216 ≤ e ≤ 2256 and e is an odd integer,
- |p-q| > 2nlen/2 - 100,
- p ≥ 2nlen/2 -1/2,
- q ≥ 2nlen/2 -1/2,
- 2(nlen/2) < d < LCM(p-1,q-1),
- e·d = 1 mod LCM(p-1,q-1).

Testing for this is covered by the ACVP/CAVP certification process

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P- 256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Testing for this is covered by the ACVP/CAVP certification process

Key Generation for Finite-Field Cryptography (FFC)

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

Cryptographic and Field Primes:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

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

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

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or

the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- $g \neq 0,1$
- q divides p-1
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

Testing for this is covered by the ACVP/CAVP certification process

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

Testing for FFC Schemes using Diffie-Hellman group 14 and/or safe-prime groups is done as part of testing in CKM.2.1.

Testing for this is covered by the ACVP/CAVP certification process

2.1.2 Cryptographic Symmetric Key Generation (FCS_CKM.1(2))

2.1.2.1 TSS Activities

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

[ST] section 6.1.2 states that the TOE invokes its own AES-256 CTR_DRBG when symmetric keys are generated.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform.

[ST] section 6.1.2 lists the mechanisms by which random bits are collected from the host platform. The TOE uses platform-collected random data as seeding for its own DRBG.

Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.

[ST] section 6.1.2 states that the vendor assumes a minimum entropy strength of 0.67 bits of entropy per collected entropy bit (understanding that the entropy sources will additionally have their own estimate for how much raw data contributed to each bit of entropy that is provided). Based on this, the vendor collects 384 bits of entropy data as an oversample to produce a minimum seed of 256 bits.

2.1.2.2 Guidance Activities

None defined.

2.1.2.3 Test Activities

None defined.

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

2.1.3.1 TSS Activities

The evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

[ST] section 6.1.3 states that 2048-bit RSA keys or ECDSA using P-256/P-384 are generated as part of IKE peer authentication.

2.1.3.2 Guidance Activities

There are no AGD EAs for this requirement.

2.1.3.3 Test Activities

If this functionality is implemented by the TSF, refer to the following EAs, depending on the TOE's claimed Base-PP:

- GPOS PP: FCS_CKM.1
- MDF PP: FCS_CKM.1
- App PP: FCS_CKM.1(1)
- MDM PP: FCS_CKM.1

2.1.4 Cryptographic Key Establishment (FCS_CKM.2)

2.1.4.1 TSS Activities

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

[ST] section 6.1.4 states that elliptic curve key establishment is used for IPsec in supports of DH groups 19 and 20 and that FFC DH group 14 key establishment is also used in support of IPsec. This is consistent with the key generation schemes that are specified.

2.1.4.2 Guidance Activities

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

[CCECG] notes in section 1 that the TOE's configuration settings are acquired from the Mobility Controller. The configuration references in section 2.2.4 and 2.2.5 of [CCECG] summarizes how to

configure the connection settings for IKEv1 and IKEv2 policies, respectively. In both of these sections, information is present on how to select the Diffie-Hellman group that is used for ISAKMP. The key generation and key establishment scheme used by the TOE is implicitly configured through specifying the Diffie-Hellman group that is used.

2.1.4.3 Test Activities

Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Testing for this is covered by the ACVP/CAVP certification process

Function Test

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

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

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

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

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

Testing for this is covered by the ACVP/CAVP certification process.

Validity Test

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

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

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

Testing for this is covered by the ACVP/CAVP certification process.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following evaluation activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTSOAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

Testing for this is covered by the ACVP/CAVP certification process.

If the TOE acts as a receiver, the following evaluation activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each

supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTSOAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

Testing for this is covered by the ACVP/CAVP certification process.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

Testing for this is covered by the ACVP/CAVP certification process.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAESPKCS1- v1_5 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses RSAES-PKCS1-v1_5.

Testing for this is covered by the ACVP/CAVP certification process.

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses Diffie-Hellman group 14.

Testing for this is covered by the ACVP/CAVP certification process.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

Testing for this is covered by the ACVP/CAVP certification process.

2.1.5 Cryptographic Key Generation Services (FCS_CKM_EXT.1)

2.1.5.1 TSS Activities

The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the generate no asymmetric cryptographic keys selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

From a review of [ST] it is clear that the TOE requires asymmetric key generation services because the implementation of IPsec requires asymmetric key generation in support of DH key establishment. Therefore, it is appropriate that the ST claims asymmetric key generation functionality. Further evaluation of this function is performed in the relevant SFRs.

2.1.5.2 Guidance Activities

None defined.

2.1.5.3 Test Activities

None defined.

2.1.6 Cryptographic Key Storage (FCS_CKM_EXT.2)

2.1.6.1 TSS Activities

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

[ST] section 6.1.6 identifies a table that lists all of the keys used by the TOE, their purpose, and where they are stored.

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

The table in [ST] section 6.1.6 notes that the TOE stores the following key data in non-volatile memory:

- RSA private key (stored in platform keystore)
- ECDSA private key (stored in platform keystore)
- Pre-shared key (uses platform-based AES and platform storage mechanism)

The platform keystores for certificates are noted as the following:

- Windows: Windows Certificate Store – claimed as a key storage mechanism in various Windows STs, e.g. section 6.4.2 in https://www.commoncriteriaportal.org/files/epfiles/2019-22-ST_lite.pdf.

- Android: Android Keystore – claimed as a key storage mechanism in https://www.niap-ccevs.org/MMO/Product/st_vid11211-st.pdf.
- Linux: Keyrings – there is no version of Linux certified against a NIAP approved Protection Profile. An EAL2 evaluation of Ubuntu 18.04 (https://www.commoncriteriaportal.org/files/epfiles/ST%20-%20Canonical%20Ubuntu%20Server%2018.04%20LTS.pdf) does not claim any key storage mechanisms in its TOE boundary. However, [App PP] lists keyrings as an acceptable key storage mechanism.

For pre-shared keys, [ST] identifies the storage locations as the registry (Windows), keychain (Linux), and app preference data (Android). The pre-shared key data is not stored in plaintext, however, because platform-based AES is used to encrypt it prior to storage. All three platform STs claim 128-bit and 256-bit AES algorithm implementations.

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

The table in [ST] section 6.1.6 notes that the TOE stores the following key data in non-volatile memory:

- RSA private key (stored in platform keystore)
- ECDSA private key (stored in platform keystore)
- Pre-shared key (uses platform-based AES and platform storage mechanism)

The only key/secret data that is manipulated by the TOE is pre-shared key data. The table references FCS_STO_EXT.1 for information on how this data is stored.

2.1.6.2 Guidance Activities

There are no AGD EAs for this requirement.

2.1.6.3 Test Activities

There are no test EAs for this requirement.

2.1.7 Cryptographic Key Destruction (FCS_CKM_EXT.4)

2.1.7.1 TSS Activities

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

[ST] section 6.1.6 lists all plaintext secret and private keys used by the TOE.

Requirement met by the platform

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

Each of the plaintext secret and private keys used by the TOE is described in [ST] section 6.1.6 along with the destruction mechanism. Specifically, the TOE platform is responsible for the destruction of pre-shared keys, as well as RSA and ECDSA private keys in persistent storage (i.e. certificates).

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

[ST] section 6.1.6 lists the supported platform key destruction mechanisms for pre-shared keys as follows:

- Windows: SecureZeroMemory() – covered as the key destruction mechanism in section 6.2.2 of https://www.commoncriteriaportal.org/files/epfiles/2019-22-ST_lite.pdf.
- Android: The Android ST (https://www.niap-ccevs.org/MMO/Product/st_vid11211-st.pdf) does not mention the specific key destruction mechanism used, but identifies that such a mechanism exists.
- Linux: The Linux ST (https://www.commoncriteriaportal.org/files/epfiles/ST%20-%20Canonical%20Ubuntu%20Server%2018.04%20LTS.pdf) does not mention the specific key destruction used, but identifies that such a mechanism exists.

Requirement met by the TOE

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

[ST] section 6.1.6 identifies several keys that reside temporarily in volatile memory and states that the TOE destroys these by overwriting once with zeroes. This section also identifies that the TOE relies on platform-based key storage for RSA and ECDSA private keys (i.e. certificates in persistent storage), but that when these are destroyed, the platform is responsible for their destruction.

2.1.7.2 Guidance Activities

There are no AGD EAs for this requirement.

2.1.7.3 Test Activities

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

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

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

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

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

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

The evaluator utilized a specially built instance of the TOE that would print out the key data prior to clearing and then after clearing. The evaluator verified that the keys were properly cleared.

2.1.8 Cryptographic Operation – Encryption/Decryption (FCS_COP.1(1))

2.1.8.1 TSS Activities

From [App PP]

None defined.

From [MOD VPNC]

If the TSF implements AES cryptography in support of both credential encryption (per FCS_STO_EXT.1) and IPsec, the evaluator shall examine the TSS to ensure that it clearly identifies the modes and key sizes that are supported for each usage of AES.

[ST] section 6.1.8 states that 128-bit AES-CBC is used for storage of pre-shared keys. This section also states that all claimed AES functionality (128-bit and 256-bit CBC and GCM) is used for IPsec.

2.1.8.2 Guidance Activities

The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.

[CCECG] notes in section 1 that the product's configuration settings are acquired from the Mobility Controller. Sections 2.2.4 and 2.2.5 describe how to configure the encryption algorithms used for ESP and IKE, which are the two situations in which AES keys are used. Therefore, guidance on the configuration of these two connection parameters is synonymous with guidance on how to configure the AES modes and key sizes.

2.1.8.3 Test Activities

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

AES-CBC Known Answer Tests

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

Testing for this is covered by the ACVP/CAVP certification process.

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

Testing for this is covered by the ACVP/CAVP certification process.

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

Testing for this is covered by the ACVP/CAVP certification process.

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

Testing for this is covered by the ACVP/CAVP certification process.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key

value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

Testing for this is covered by the ACVP/CAVP certification process.

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

Testing for this is covered by the ACVP/CAVP certification process.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation. AES-CBC Monte Carlo Tests The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3- tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

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

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial.

This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

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

Testing for this is covered by the ACVP/CAVP certification process.

Added by TD0598.

AES-CCM Tests

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

Keys: All supported and selected key sizes (e.g., 128, 256 bits).

Associated Data: Two or three values for associated data length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.

Payload: Two values for payload length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported payload lengths.

Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes.

Tag: All supported tag lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test

For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Payload Test

For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Nonce Test

For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Tag Test

For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Decryption-Verification Process Test

To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

This testing is not applicable to the TOE; the TOE does not claim AES-CCM.

Added by TD0598.

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

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

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

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

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

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

Test 2: Multi-Block Message Test

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

shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

Test 3: Monte-Carlo Test

For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode
Input: PT, Key
for i = 1 to 1000:
CT[i] = AES-ECB-Encrypt(Key, PT)
PT = CT[i]

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

This testing is not applicable to the TOE; the TOE does not claim AES-CTR.

AES-GCM Monte Carlo Tests

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

- 128 bit and 256 bit keys
- Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

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

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

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall

compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Testing for this is covered by the ACVP/CAVP certification process.

AES-XTS Tests

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

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

Testing for this is covered by the ACVP/CAVP certification process.

2.1.9 Cryptographic Operation – Hashing (FCS_COP.1(2))

2.1.9.1 TSS Activities

The evaluator shall check that the association of the hash function with other application cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

[ST] section 6.1.9 states that SHA-1 is used in ESP, SHA-1/256/384 are used for IKE, and SHA-1 is used for code signing.

2.1.9.2 Guidance Activities

None defined.

2.1.9.3 Test Activities

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

Testing for this is covered by the ACVP/CAVP certification process.

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

Testing for this is covered by the ACVP/CAVP certification process.

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

Testing for this is covered by the ACVP/CAVP certification process.

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

Testing for this is covered by the ACVP/CAVP certification process.

Test 2: Short Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Testing for this is covered by the ACVP/CAVP certification process.

Test 3: Selected Long Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Testing for this is covered by the ACVP/CAVP certification process.

Test 4: Selected Long Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Testing for this is covered by the ACVP/CAVP certification process.

Test 5: Pseudorandomly Generated Messages Test This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

Testing for this is covered by the ACVP/CAVP certification process.

2.1.10 Cryptographic Operation – Signing (FCS_COP.1(3))

2.1.10.1 TSS Activities

None defined.

2.1.10.2 Guidance Activities

None defined.

2.1.10.3 Test Activities

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

ECDSA Algorithm Tests

- **Test 1:** ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.
- **Test 2:** ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

- **Test 1:** Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.
- **Test 2:** Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

Testing for this is covered by the ACVP/CAVP certification process.

2.1.11 Cryptographic Operation – Keyed-Hash Message Authentication (FCS_COP.1(4))

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

2.1.11.1 TSS Activities

None defined.

2.1.11.2 Guidance Activities

None defined.

2.1.11.3 Test Activities

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

Testing for this is covered by the ACVP/CAVP certification process.

2.1.12 IPsec (FCS_IPSEC_EXT.1)

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

The TOE is required to use the IPsec protocol to establish connections used to communicate with an IPsec peer.

In addition to the TSS EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the TOE boundary includes a general-purpose operating system or mobile device, the evaluator shall examine the TSS to ensure that it describes whether the VPN client capability is architecturally integrated with the platform itself or whether it is a separate executable that is bundled with the platform.

The TOE is a software application and does not include a general-purpose operating system or mobile device in its evaluation boundary.

In addition to the Operational Guidance EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall perform the following:

If the configuration of the IPsec behavior is from an environmental source, most notably a VPN gateway (e.g through receipt of required connection parameters from a VPN gateway), the evaluator shall ensure that the operational guidance contains any appropriate information for ensuring that this configuration can be properly applied.

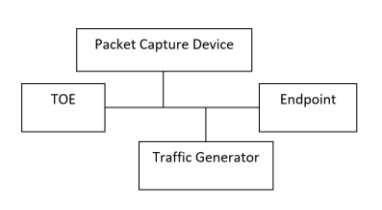
Note in this case that the implementation of the IPsec protocol must be enforced entirely within the TOE boundary; i.e. it is not permissible for a software application TOE to be a graphical front-end for IPsec functionality implemented totally or in part by the underlying OS platform. The behavior referenced here is for the possibility that the configuration of the IPsec connection is initiated from outside the TOE, which is permissible so long as the TSF is solely responsible for enforcing the configured behavior. However, it is allowable for the TSF to rely on low-level platform-provided

networking functions to implement the SPD from the client (e.g., enforcement of packet routing decisions).

[CCECG] notes that the TOE's VPN client configuration is dependent on configuration parameters received from a VPN gateway, specifically an Aruba Mobility Controller. It references separate Aruba documentation that describes the process for configuring the VPN gateway and applying those settings to the TOE. Specifically, [CCECG] section 2.2 and the various subsections describe how to configure IPsec.

As a prerequisite for performing the Test EAs for the individual FCS_IPSEC_EXT.1 elements below, the evaluator shall do the following:

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



Note that the evaluator shall perform all tests using the VPN client and a representative sample of platforms listed in the ST (for TOEs that claim to support multiple platforms).

2.1.12.1 TSS Activities

FCS_IPSEC_EXT.1.1

The evaluator shall examine the TSS and determine that it describes how the IPsec capabilities are implemented.

[ST] section 6.1.12 states that IPsec is implemented cryptographically using the Aruba Common Crypto Module (ACCM) and implemented by the TOE application itself (i.e. it does not rely on the platform), and that the TOE interacts with the platform network stack using platform APIs.

FCS_IPSEC_EXT.1.1

If the TOE is a standalone software application, the evaluator shall ensure that the TSS asserts that all IPsec functionality is implemented by the TSF. The evaluator shall also ensure that the TSS identifies what platform functionality the TSF relies upon to support its IPsec implementation, if any (e.g. does it invoke cryptographic primitive functions from the platform's cryptographic library, enforcement of packet routing decisions by low-level network drivers).

[ST] section 6.1.12 states that the TOE implements IPsec as a standalone application that is not part of its underlying OS platform, and that it uses sockets to interact with the platform networking stack.

FCS_IPSEC_EXT.1.1

If the TOE is part of a general-purpose desktop or mobile operating system, the evaluator shall ensure that the TSS describes at a high level the architectural relationship between the VPN client portion of the TOE and the rest of the TOE (e.g. is the VPN client an integrated part of the OS or is it a standalone

executable that is bundled into the OS package). If the SPD is implemented by the underlying platform in this case, then the TSS describes how the client interacts with the platform to establish and populate the SPD, including the identification of the platform's interfaces that are used by the client.

The TOE is not part of a general-purpose desktop or mobile operating system; this evaluation activity is N/A.

FCS_IPSEC_EXT.1.1
In all cases, the evaluator shall also ensure that the TSS describes how the client interacts with the network stack of the platform(s) on which it can run (e.g., does the client insert itself within the stack via kernel mods, does the client simply invoke APIs to gain access to network services).

[ST] section 6.1.12 states that the TOE relies on platform APIs to interact with the network stacks of the respective platforms using sockets.

FCS_IPSEC_EXT.1.1
The evaluator shall ensure that the TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how the available rules and actions form the SPD using terms defined in RFC 4301 such as BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

[ST] section 6.1.12 states that the TOE implements minimal SPD rules and that the administrator must ensure that the VPN gateway configuration establishes a DISCARD rule such that anything that is not processed as BYPASS or PROTECT rule is discarded. This section also states that BYPASS and PROTECT rules are implicitly configured through the use of split tunneling. It is understood by the evaluator that when split tunneling is enabled, any valid traffic to the tunneled network would be processed as PROTECT while any valid traffic to any other network would be processed as BYPASS.

FCS_IPSEC_EXT.1.1
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. 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.

The TOE is not responsible for enforcing SPD rules; the SPD rules are enforced by configuration at the VPN gateway. From a user perspective, traffic will be PROTECTed if transmitted through the VPN (i.e. in all cases where split tunneling is disabled, and when traffic is bound for the tunneled network when split tunneling is enabled), BYPASSed if split tunneling is enabled and traffic is not bound for the tunneled network, and DISCARDed if the traffic is found to be in violation of a firewall rule. Regardless of whether a split tunnel is enforced or not, all traffic originating from the client is passed through the TOE to the VPN gateway once an initial connection to it is established.

FCS_IPSEC_EXT.1.2

The evaluator shall check the TSS to ensure it states that the VPN can be established to operate in tunnel mode and/or transport mode (as selected).

[ST] section 6.1.12 states that the TOE operates in tunnel mode only, consistent with the SFR claim.

FCS_IPSEC_EXT.1.3

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

[ST] section 6.1.12 states that packets are processed against the SPD rules that are defined implicitly through the configuration and connection of a VPN session. Specifically, PROTECT and BYPASS rules are implicit for allowed traffic, where PROTECT rules apply in all cases except when a split tunnel is defined and the traffic is not bound for a destination in the tunneled network. This section states that the DISCARD rule is enforced via firewall rules. It is understood from this that the expectation is that the VPN gateway has a firewall policy configured to discard all traffic that doesn't match any traffic that is explicitly authorized.

FCS_IPSEC_EXT.1.4

The evaluator shall examine the TSS to verify that the algorithms AES-GCM-128 and AES-GCM-256 are implemented. If the ST author has selected either AES-CBC-128 or AES-CBC-256 in the requirement, then the evaluator verifies the TSS describes these as well. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in the relevant iteration of FCS_COP.1 from the Base-PP that applies to keyed-hash message authentication.

[ST] section 6.1.12 identifies AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES-CBC-256 as the supported ESP algorithms, consistent with the SFR claim. This section also identifies HMAC-SHA1 as the authentication algorithm.

FCS_IPSEC_EXT.1.5

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. If IKEv1 is implemented, the evaluator shall verify that the TSS indicates whether or not XAUTH is supported, and that aggressive mode is not used for IKEv1 Phase 1 exchanges (i.e. only main mode is used). It may be that these are configurable options.

[ST] section 6.1.12 states that IKEv1 and IKEv2 are implemented. For IKEv1, this section also states that aggressive mode is not supported and that XAUTH is supported.

FCS_IPSEC_EXT.1.6

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

[ST] section 6.1.1.12 states that AES-CBC-128 and AES-CBC-256 are supported for the IKE encryption algorithms, consistent with the SFR claims.

FCS_IPSEC_EXT.1.7

There are no TSS EAs for this requirement.

N/A

FCS_IPSEC_EXT.1.8
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.

[ST] section 6.1.12 states that the TOE supports DH groups 14, 19, and 20, consistent with the claims made in the SFR. This section also states that there is no negotiation of DH groups because the chosen connection policy will only have one DH group associated with it.

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

[ST] section 6.1.12 states that the value of x is generated by the TOE's DRBG, and that the number of bits depends on the DH group: 224 bits for group 14, 256 bits for group 19, and 384 bits for group 20. This is sufficient to ensure that the likelihood of a repeated nonce over the lifetime of an SA is less than 1^(n/2) where n is the number of bits.

FCS_IPSEC_EXT.1.10
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.9

N/A

FCS_IPSEC_EXT.1.11
The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication.

[ST] section 6.1.12 states that both RSA and ECDSA certificates are supported for peer authentication.

FCS_IPSEC_EXT.1.11
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 description in the TSS shall also indicate how pre-shared key establishment is accomplished depending on whether the TSF can generate a pre-shared key, accept a pre-shared key, or both.

[ST] section 6.1.12 states that pre-shared keys are supported, and that they are configured on the VPN gateway. This is consistent with the selection in FIA_PSK_EXT.1.3 that the TSF can only accept a pre-shared key and not generate it.

This section also states that pre-shared keys can only be used when IKEv1 is configured; IKEv2 does not support pre-shared keys.

FCS_IPSEC_EXT.1.11
The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include whether the certificate presented identifier is compared to the ID payload presented identifier, which field(s) of the certificate are used as the

presented identifier (DN, Common Name, or SAN) and, if multiple fields are supported, the logical order comparison. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate.

[ST] section 6.1.12 states that the reference identifier of the certificate is validated by comparison of the certificate DN to the expected DN for the peer. No other fields are used.

FCS_IPSEC_EXT.1.12, FCS_IPSEC_EXT.1.13
None defined.

N/A

FCS_IPSEC_EXT.1.14
The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

[ST] section 6.1.12 states that only AES is used for the IKE and ESP exchanges. From this it is understood that the key strength is equal to the number of bits of the key. This section also states that the TOE is configured by the environmental VPN gateway, so it is necessary to ensure through configuration that the security strength of the IKE_SA is greater than or equal to that of the CHILD_SA.

2.1.12.2 Guidance Activities

FCS_IPSEC_EXT.1.1
The evaluator shall examine the operational guidance to verify it describes how the SPD is created and configured. If there is an administrative interface to the client, then the guidance describes how the administrator specifies rules for processing a packet. The description includes all three cases - a rule that ensures packets are encrypted/decrypted, dropped, and allowing a packet to flow in plaintext. The evaluator shall determine that the description in the operational guidance is consistent with the description in the TSS, 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. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

N/A – the client is configured by an external application.

FCS_IPSEC_EXT.1.1
If the client is configured by an external application, such as the VPN gateway, then the operational guidance should indicate this and provide a description of how the client is configured by the external application. The description should contain information as to how the SPD is established and set up in an unambiguous fashion. The description should also include what is configurable via the external application, how ordering of entries may be expressed, as well as the impacts that ordering of entries may have on the packet processing.

[CCECG] section 2.2.3 references how PROTECT and BYPASS rules can be implicitly enabled through VPN gateway configuration, and that an explicit DISCARD rule needs to be defined on the gateway. This

section also discusses the order priority of rule application, specifically in regards to resolving what would otherwise be potential conflicts between rules.

FCS_IPSEC_EXT.1.1
In either case, the evaluator ensures the description provided In the TSS is consistent with the capabilities and description provided in the operational guidance.

Both [ST] and [CCECG] discuss the SPD being enforced by an external VPN gateway and dependent on split tunneling being configured.

FCS_IPSEC_EXT.1.2
The evaluator shall confirm that the operational guidance contains instructions on how to configure the connection in each mode selected.

[CCECG] notes in section 1 that the product's configuration settings are acquired from the Mobility Controller. The configuration references in sections 2.2.4 and 2.2.5 identify how the use of tunnel mode is forced through configuration to ensure that transport mode is not used.

FCS_IPSEC_EXT.1.2
If both transport mode and tunnel mode are implemented, the evaluator shall review the operational guidance to determine how the use of a given mode is specified.

N/A; the TOE does not support transport mode.

FCS_IPSEC_EXT.1.3
The evaluator shall check that the operational guidance provides instructions on how to construct or acquire the SPD and uses the guidance to configure the TOE for the following test.

[CCECG] states that the TOE will acquire the SPD when a connection attempt to a given VPN profile is made. The VPN gateway will define an SPD that may use an access-list to define PROTECT/BYPASS/DISCARD rules, and its connection profile will either have split tunneling enabled or disabled, which determines whether non-tunneled network traffic gets processed using a PROTECT or BYPASS rule.

FCS_IPSEC_EXT.1.4
The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

[CCECG] notes in section 1 that the product's configuration settings are acquired from the Mobility Controller. The configuration instructions in sections 2.2.4 and 2.2.5 describe how to configure the encryption algorithms used for ESP, which are configured in the Transforms algorithms that are specified in the IPsec Dynamic Map.

FCS_IPSEC_EXT.1.5
The evaluator shall check 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 test below. If XAUTH is implemented, the evaluator shall verify that the operational guidance provides instructions on how it is enabled or disabled.

The "Configuring VIA Settings in ArubaOS 8.x" section of [User] has instructions under the "Creating VIA Connection Profiles" header for how to create a connection profile. Table 4 in this section lists the connection profile settings which includes the "Enable IKEv2" checkbox to specify the version of IKE being used.

The "Authentication Methods" section of [User] includes a chapter called "Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features." This chapter identifies how to enable XAUTH for IKEv1.

FCS_IPSEC_EXT.1.5
If the TOE supports IKEv1, the evaluator shall verify that the operational guidance either asserts that only main mode is used for Phase 1 exchanges, or provides instructions for disabling aggressive mode.

[CCECG] notes in section 1 that the product's configuration settings are acquired from the Mobility Controller. The configuration instructions for IKEv1 in section 2.2.4 notes that the "crypto-local isakmp disable-aggressive-mode" command must be used to ensure that aggressive mode is disabled. Section 2.2, which is the lead-in section for all IPsec related configuration guidance, also notes that aggressive mode is not supported for IKEv1, so the existence of this restriction should be clear to the administrator.

FCS_IPSEC_EXT.1.6
The evaluator checks the operational guidance to ensure it provides instructions on how the TOE is configured to use the algorithms selected in this component and whether this is performed through direct configuration, defined during initial installation, or defined by acquiring configuration settings from an environmental component.

The "Authentication Methods Supported in VIA" section of [User] includes a chapter called "Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features." This chapter identifies configure the supported IKE version on the VPN gateway.

FCS_IPSEC_EXT.1.7
The evaluator shall check the operational guidance to ensure it provides instructions on how the TOE configures the values for SA lifetimes. In addition, the evaluator shall check that the guidance has the option for either the Administrator or VPN Gateway to configure Phase 1 SAs if time-based limits are supported. Currently there are no values mandated for the number of packets or number of bytes, the evaluator shall simply check the operational guidance to ensure that this can be configured if selected in the requirement.

The "Authentication Methods Supported in VIA" section of [User] includes a chapter called "Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features." This chapter identifies how to configure the IKE and IPsec rekey intervals. [CCECG] section 2.2.4 also describes the configuration of the authentication method.

FCS_IPSEC_EXT.1.8
There are no AGD EAs for this requirement.

N/A

FCS_IPSEC_EXT.1.9

There are no AGD EAs for this requirement.

N/A

FCS_IPSEC_EXT.1.10
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.9

N/A

FCS_IPSEC_EXT.1.11
The evaluator shall check that the operational guidance describes how pre-shared keys are to be generated and established.

The "Authentication Methods Supported in VIA" section of [User] includes a chapter called "Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features." This chapter identifies how to specify a pre-shared key. This information is also covered in section 2.2.4 of [CCECG].

FCS_IPSEC_EXT.1.11
The evaluator ensures the operational guidance describes how to set up the TOE to use the cryptographic algorithms RSA and/or ECDSA.

Section 2.2.4 of [CCECG] describes how to use the mobility controller to configure RSA versus ECDSA encryption. The "Authentication Methods Supported in VIA" section of [User] also includes a chapter called "Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features," which also describes how to configure the authentication algorithm that is used.

FCS_IPSEC_EXT.1.11
In order to construct the environment and configure the TOE for the following tests, the evaluator will ensure that the operational 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 as a trusted CA.

Section 2.3 of [CCECG] describes the process for loading certificates into the TOE, which includes guidance for specifying a trusted CA certificate as a trusted CA.

FCS_IPSEC_EXT.1.11
The evaluator shall also ensure that the operational guidance includes the configuration of the reference identifier(s) for the peer.

[CCECG] section 2.3 describes the process for certificate checking. This section notes the VPN gateway configuration instructions that will cause the TOE to check the DN of the presented certificate when the initial connection to the gateway is requested.

FCS_IPSEC_EXT.1.12, FCS_IPSEC_EXT.1.13
None defined.

N/A

FCS_IPSEC_EXT.1.14
There are no AGD EAs for this requirement.

N/A

2.1.12.3 Test Activities

FCS_IPSEC_EXT.1.1

Depending on the implementation, the evaluator may be required to use a VPN gateway or some form of application to configure the client. For Test 2, the evaluator is required to choose an application that allows for the configuration of the full set of capabilities of the VPN client (in conjunction with the platform). For example, if the client provides a robust interface that allows for specification of wildcards, subnets, etc., it is unacceptable for the evaluator to choose a VPN Gateway that only allows for specifying a single fully qualified IP addresses in the rule.

The evaluator shall perform the following tests:

FCS_IPSEC_EXT.1.1

Test 1: The evaluator shall configure an SPD on the client that is capable of the following: dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the client with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule. The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behavior: appropriate packets were dropped, allowed through without modification, was encrypted by the IPsec implementation.

The evaluator configured the TOE to encrypt traffic by specifying a target VPN gateway to connect to and observing that traffic bound for a host in the VPN is encrypted. The evaluator configured the TOE to allow a packet to flow in plaintext by configuring the TOE to use a split tunnel, connecting to a target VPN gateway, and observing that traffic bound for a host outside the VPN is not encrypted. The TOE enforces an implicit deny operation so the evaluator tested this by configuring the TOE not to use a split tunnel, connecting to a target VPN gateway, and observing that traffic bound for a host outside the VPN is discarded.

FCS_IPSEC_EXT.1.1

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 TSS and operational guidance. 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, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the operational guidance.

The TOE enforces a rudimentary SPD where traffic is PROTECTed if bound for a host in the VPN and either BYPASSed or DISCARDed if not, depending on whether or not a split tunnel is enabled. No more granular configuration of the SPD is possible. The evaluator verified as part of performing Test 1 that this SPD is enforced.

FCS_IPSEC_EXT.1.2

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

FCS_IPSEC_EXT.1.2

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 gateway to operate in tunnel mode. The evaluator configures the TOE and the VPN gateway to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the client to connect to the VPN GW peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

The TOE only supports tunnel mode and the evaluator observed that the TOE was capable of constructing an IPSEC channel that utilized tunnel mode.

FCS_IPSEC_EXT.1.2

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 an IPsec peer to accept IPsec connections using transport mode. The evaluator configures the TOE and the endpoint device 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 remote endpoint. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

N/A, the TOE does not claim to support transport mode.

FCS_IPSEC_EXT.1.2

Test 3 [conditional]: If both tunnel mode and transport mode are selected, the evaluator shall perform both Test 1 and Test 2 above, demonstrating that the TOE can be configured to support both modes.

N/A, the TOE only claims to support tunnel mode.

FCS_IPSEC_EXT.1.2

Test 4 [conditional]: If both tunnel mode and transport mode are selected, the evaluator shall modify the testing for FCS_IPSEC_EXT.1 to include the supported mode for SPD PROTECT entries to show that they only apply to traffic that is transmitted or received using the indicated mode.

N/A, the TOE only claims to support tunnel mode.

FCS_IPSEC_EXT.1.3

The evaluator shall perform the following test:

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

The TOE enforces a rudimentary SPD where traffic is PROTECTed if bound for a host in the VPN and either BYPASSed or DISCARDed if not, depending on whether or not a split tunnel is enabled. No more granular configuration of the SPD is possible. The evaluator verified as part of performing FCS_IPSEC_EXT.1.1 Test 1 that this SPD is enforced.

FCS_IPSEC_EXT.1.4

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

The evaluator observed that the TOE was capable of utilizing each of the selected algorithms to encrypt the phase 2 channel (ESP).

FCS_IPSEC_EXT.1.5

Test 1: The evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 7296, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed. If the TOE supports IKEv1 with or without XAUTH, the evaluator shall verify that this test can be successfully repeated with XAUTH enabled and disabled in the manner specified by the operational guidance. If the TOE only supports IKEv1 with XAUTH, the evaluator shall verify that connections not using XAUTH are unsuccessful. If the TOE only supports IKEv1 without XAUTH, the evaluator shall verify that connections using XAUTH are unsuccessful.

The evaluator observed that the TOE is capable of handling NAT traversal for IKEv2. The evaluator observed that the gateway only allows for XAUTH to be enabled and that the TOE cannot be configured differently from the gateway since the gateway is the authoritative source of its configuration.

FCS_IPSEC_EXT.1.5

Test 2 [conditional]: If the TOE supports IKEv1, the evaluator shall perform any applicable operational guidance steps to disable the use of aggressive mode and then attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator shall show that the TOE will reject a VPN gateway from initiating an IKEv1 Phase 1 connection in aggressive mode. The evaluator should then show that main mode exchanges are supported.

The evaluator observed that the gateway only supports main mode for IKEv1. The evaluator attempted to modify the configuration of the TOE through editing of locally-stored configuration settings to attempt to establish an aggressive mode connection. The evaluator observed either that settings could not be modified outside the TOE without causing the TOE to fail to start (Windows) or that initial HTTP connection to the gateway would cause the gateway to detect the mismatch and update the TOE's local configuration before an IPsec connection is attempted (Android, Linux). In all cases it was not possible to attempt an IPsec connection using aggressive mode.

FCS_IPSEC_EXT.1.6

The evaluator shall use the operational guidance to configure the TOE (or to configure the Operational Environment to have the TOE receive configuration) to perform the following test for each ciphersuite selected:

Test 1: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept

the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation. The evaluator will confirm that the connection is successful by confirming that data can be passed through the connection once it is established. For example, the evaluator may connect to a webpage on the remote network and verify that it can be reached.

The evaluator observed that the IKE phase 1 connection could be completed using each of the selected cipher suites.

FCS_IPSEC_EXT.1.7

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

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

FCS_IPSEC_EXT.1.7

Test 1 [conditional]: The evaluator shall configure a maximum lifetime in terms of the # of packets (or bytes) allowed following the operational guidance. The evaluator shall establish an SA and determine that once the allowed # of packets (or bytes) through this SA is exceeded, the connection is closed.

N/A, The TOE does not claim to support maximum lifetimes by the number of packets or bytes.

FCS_IPSEC_EXT.1.7

Test 2 [conditional]: The evaluator shall construct a test where a Phase 1 SA is established and attempted to be maintained for more than 24 hours before it is renegotiated. The evaluator shall observe that this SA is closed or renegotiated in 24 hours or less. If such an action requires that the TOE be configured in a specific way, the evaluator shall implement tests demonstrating that the configuration capability of the TOE works as documented in the operational guidance.

The TOE is only capable of communicating with an Aruba Mobility Controller which is itself validated. Thus, the controller will enforce these controls onto the TOE. The evaluator demonstrated that when a connection profile with a specified Phase 1 SA timeout limit is loaded onto the TOE, the TOE will abide by this limit and a rekey will occur when the specified length of time has elapsed.

FCS_IPSEC_EXT.1.7

Test 3 [conditional]: The evaluator shall perform a test similar to Test 2 for Phase 2 SAs, except that the lifetime will be 8 hours or less instead of 24 hours or less.

The TOE is only capable of communicating with an Aruba Mobility Controller which is itself validated. Thus, the controller will enforce these controls onto the TOE. The evaluator demonstrated that when a connection profile with a specified Phase 2 SA timeout limit is loaded onto the TOE, the TOE will abide by this limit and a rekey will occur when the specified length of time has elapsed.

FCS_IPSEC_EXT.1.7

Test 4 [conditional]: If a fixed limit for IKEv1 SAs is supported, the evaluator shall establish an SA and observe that the connection is closed after the fixed traffic and/or time value is reached.

N/A, The TOE does not claim to support a fixed limit.

FCS_IPSEC_EXT.1.8

The evaluator shall perform the following test:

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

The evaluator verified that the TOE is capable of using each of the specified DH groups for each IKE protocol.

FCS_IPSEC_EXT.1.9

There are no test EAs for this requirement.

FCS_IPSEC_EXT.1.10

EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.9.

FCS_IPSEC_EXT.1.11

For efficiency's sake, the testing that is performed here has been combined with the testing for FIA_X509_EXT.2 and FIA_X509_EXT.3 (for IPsec connections and depending on the Base-PP), FCS_IPSEC_EXT.1.12, and FCS_IPSEC_EXT.1.13. The following tests shall be repeated for each peer authentication protocol selected in the FCS_IPSEC_EXT.1.11 selection above:

FCS_IPSEC_EXT.1.11

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. Alternatively, the evaluator may import to the TOE a previously generated private key and corresponding certificate.

The TOE is not capable of generating a CSR, thus the evaluator observed that a previously generated certificate and key could be loaded and used by the TOE. Specifically, the evaluator observed that in Linux, the certificate is loaded directly into the TOE using its own management interface, while the Windows and Android versions of the TOE rely on the platform-provided certificate store as their repository for available certificates.

FCS_IPSEC_EXT.1.11

Test 2: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.

The evaluator observed that the TOE could be configured to use an externally generated Certificate and key to complete a connection with the Gateway.

FCS_IPSEC_EXT.1.11

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 current version of the PP-Module, 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.

The evaluator verified that the TOE was capable of validating the Certificate status, if one is present, of a certificate presented by the gateway prior to completing the connection.

FCS_IPSEC_EXT.1.11

Test 4 [conditional]: The evaluator shall generate a pre-shared key and use it, as indicated in the operational guidance, to establish an IPsec connection with the VPN GW peer. If the generation of the pre-shared key is supported, the evaluator shall ensure that establishment of the key is carried out for an instance of the TOE generating the key as well as an instance of the TOE merely taking in and using the key.

The Evaluator observed that the gateway was able to configure the TOE to use pre-shared key's and that a connection could be successfully established using the provided key.

FCS_IPSEC_EXT.1.11

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

FCS_IPSEC_EXT.1.11

Test 5: For each field of the certificate supported for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds.

N/A, the TOE only uses the DN identifier type, which is explicitly excluded from this testing.

FCS_IPSEC_EXT.1.11

Test 6: For each field of the certificate support for comparison, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to not match the field in the peer's presented certificate and shall verify that the IKE authentication fails.

N/A, the TOE only uses the DN identifier type, which is explicitly excluded from this testing.

FCS_IPSEC_EXT.1.11

Test 7 [conditional]: If, according to the TSS, the TOE supports both Common Name and SAN certificate fields and uses the preferred logic outlined in the Application Note, the tests above with the Common Name field shall be performed using peer certificates with no SAN extension. Additionally, the evaluator shall configure the peer's reference identifier on the TOE to not match the SAN in the peer's presented

certificate but to match the Common Name in the peer's presented certificate, and verify that the IKE authentication fails.

N/A, the TOE only uses the DN identifier type, which is explicitly excluded from this testing.

FCS_IPSEC_EXT.1.11
Test 8 [conditional]: If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds. To demonstrate a bit-wise comparison of the DN, the evaluator shall change a single bit in the DN (preferably, in an Object Identifier (OID) in the DN) and verify that the IKE authentication fails. To demonstrate a comparison of DN values, the evaluator shall change any one of the four DN values and verify that the IKE authentication fails.

N/A, the TOE only uses the DN identifier type, which is explicitly excluded from this testing.

FCS_IPSEC_EXT.1.11
Test 9 [conditional]: If the TOE supports both IPv4 and IPv6 and supports IP address identifier types, the evaluator must repeat test 1 and 2 with both IPv4 address identifiers and IPv6 identifiers. Additionally, the evaluator shall verify that the TOE verifies that the IP header matches the identifiers by setting the presented identifiers and the reference identifier with the same IP address that differs from the actual IP address of the peer in the IP headers and verifying that the IKE authentication fails.

N/A, the TOE only uses the DN identifier type, which is explicitly excluded from this testing.

FCS_IPSEC_EXT.1.11
Test 10 [conditional]: If, according to the TSS, the TOE performs comparisons between the peer's ID payload and the peer's certificate, the evaluator shall repeat the following test for each combination of supported identifier types and supported certificate fields (as above). The evaluator shall configure the peer to present a different ID payload than the field in the peer's presented certificate and verify that the TOE fails to authenticate the IKE peer.

N/A, the TOE only uses the DN identifier type, which is explicitly excluded from this testing.

FCS_IPSEC_EXT.1.12, FCS_IPSEC_EXT.1.13
EAs for this element are tested through EAs for FCS_IPSEC_EXT.1.11.

FCS_IPSEC_EXT.1.14
The evaluator follows the guidance to configure the TOE to perform the following tests.

FCS_IPSEC_EXT.1.14
Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

This test was performed in conjunction with FCS_IPSEC_EXT.1.4. That test demonstrated the use of all of the algorithms and hash functions identified in the [ST].

FCS_IPSEC_EXT.1.14
Test 2 [conditional]: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

The evaluator found that the gateway could be configured to present the configuration values where the phase 2 would be stronger than the phase 1. The evaluator observed that when this was attempted that the TOE rejected the configuration and would not complete the connection.

FCS_IPSEC_EXT.1.14
Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

The evaluator found that the gateway could not be configured to present a non-selected weak encryption algorithm to the TOE. Per the testing done for FCS_IPSEC_EXT.1.5, the evaluator demonstrated that local attempts to modify the gateway-issued configuration settings cannot result in the TOE attempting an IPsec connection using those settings; such modifications either cause the TOE to fail to start until a fresh profile is sent to it by the gateway, or the gateway will detect the mismatched configuration and will override the TOE's local configuration with the one issued by the gateway. In either case, any attempt to modify the configuration to use a non-supported IKE algorithm will be unsuccessful due to this design.

FCS_IPSEC_EXT.1.14
Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

The evaluator found that the gateway could not be configured to present a non-selected weak encryption algorithm to the TOE and thus this is implicitly met as it cannot be configured. And as with the case as above, no attempt can be made to locally configure the client to subvert the configuration settings on the gateway.

2.1.13 Random Bit Generation Services (FCS_RBG_EXT.1)

2.1.13.1 TSS Activities

If use no DRBG functionality is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

This selection was not chosen so this activity is N/A to the TOE.

If implement DRBG functionality is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

The evaluator observed that FCS_RBG_EXT.2 is appropriately claimed.

If invoke platform-provided DRBG functionality is selected, the evaluator performs the following activities. The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

This selection was not chosen so this activity is N/A to the TOE.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

This selection was not chosen so this activity is N/A to the TOE.

2.1.13.2 Guidance Activities

None defined.

2.1.13.3 Test Activities

Modified by TD0416

If invoke platform-provided DRBG functionality is selected, the following tests shall be performed:

The TOE claims to Implement the DRBG functionality thus, this testing is N/A.

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The TOE claims to Implement the DRBG functionality thus, this testing is N/A.

The following are the per-platform list of acceptable APIs:

For Android: The evaluator shall verify that the application uses at least one of javax.crypto.KeyGenerator class or the java.security.SecureRandom class or /dev/random or /dev/urandom.

The TOE claims to Implement the DRBG functionality thus, this testing is N/A.

For Windows: The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In

future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

The TOE claims to Implement the DRBG functionality thus, this testing is N/A.

For iOS: The evaluator shall verify that the application invokes either SecRandomCopyBytes, CCRandomGenerateBytes or CCRandRandomCopyBytes, or uses /dev/random directly to acquire random.

N/A, the TOE does not claim this platform.

For Linux: The evaluator shall verify that the application collects random from /dev/random or /dev/urandom.

The TOE claims to Implement the DRBG functionality thus, this testing is N/A.

For Solaris: The evaluator shall verify that the application collects random from /dev/random.

N/A, the TOE does not claim this platform.

For macOS: The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random. If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

N/A, the TOE does not claim this platform.

2.1.14 Random Bit Generation from Application (FCS_RBG_EXT.2)

2.1.14.1 TSS Activities

FCS_RBG_EXT.2.1
None defined.

N/A

FCS_RBG_EXT.2.2
Documentation shall be produced - and the evaluator shall perform the activities – in accordance with Appendix D – Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

A proprietary Entropy Analysis Report (EAR) was provided as part of the product submission.

2.1.14.2 Guidance Activities

FCS_RBG_EXT.2.1, FCS_RBG_EXT.2.2
None defined.

2.1.14.3 Test Activities

FCS_RBG_EXT.2.1
The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

FCS_RBG_EXT.2.1
Implementations Conforming to FIPS 140-2 Annex C.
The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.1
Test 1: The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.1
Test 2: The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.1
Implementations Conforming to NIST Special Publication 800-90A
Test 1: 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.

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.1
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 800-90A).

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.1

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.

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.1

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 Derivation Function 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 less then or equal to 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.

Testing for this is covered by the ACVP/CAVP certification process.

FCS_RBG_EXT.2.2

In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates.

Testing for this is covered by the ACVP/CAVP certification process.

2.1.15 Storage of Credentials (FCS_STO_EXT.1)

2.1.15.1 TSS Activities

The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

[ST] section 6.1.14 lists two types of credentials that the TOE stores: certificates and IKEv1 pre-shared keys. Certificates are used for establishment of IPsec communications, as are pre-shared keys (if configured by the gateway for use as an authenticator).

This section states that pre-shared keys are both protected using AES, regardless of the platform. For certificates, all three platform versions of the TOE use platform-provided storage mechanisms (Windows Certificate Store, Linux keyrings, Android keystore).

2.1.15.2 Guidance Activities

None defined.

2.1.15.3 Test Activities

For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1(1) or conditioned according to FCS_CKM.1.1(1) and FCS_CKM.1(3). For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

For Android: The evaluator shall verify that the application uses the Android KeyStore or the Android KeyChain to store certificates.

The evaluator verified that the TOE used the Android KeyStore for storage of certificates.

For Windows: The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

The Evaluator verified that the TOE used the windows certificate store to store credentials.

For iOS: The evaluator shall verify that all credentials are stored within a Keychain.

N/A, the TOE does not claim this platform.

For Linux: The evaluator shall verify that all keys are stored using Linux keyrings.

The evaluator verified that the TOE stored keys using Linux keyrings.

For Solaris: The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).

N/A, the TOE does not claim this platform.

For macOS: The evaluator shall verify that all credentials are stored within Keychain.

N/A, the TOE does not claim this platform.

2.2 User Data Protection (FDP)

2.2.1 Encryption of Sensitive Application Data (FDP_DAR_EXT.1)

2.2.1.1 TSS Activities

The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application.

The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS.

If not store any sensitive data is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory.

The evaluator shall also ensure that this is consistent with the filesystem test below.

[ST] section 6.2.1 states that the only 'sensitive data' is the credential data protected by the mechanisms claimed in FCS_STO_EXT.1.

2.2.1.2 Guidance Activities

None defined.

2.2.1.3 Test Activities

Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1.

The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

If leverage platform-provided functionality is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis.

For Android: The evaluator shall inspect the TSS and verify that it describes how files containing sensitive data are stored with the MODE_PRIVATE flag set.

N/A, The TOE only stores data according to FCS_STO_EXT.1.

For Windows: The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

N/A, The TOE only stores data according to FCS_STO_EXT.1.

For iOS: The evaluator shall inspect the TSS and ensure that it describes how the application uses the Complete Protection, Protected Unless Open, or Protected Until First User Authentication Data Protection Class for each data file stored locally.

N/A, the TOE does not claim this platform.

For Linux: The Linux platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

N/A, The TOE only stores data according to FCS_STO_EXT.1.

For Solaris: The Solaris platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

N/A, the TOE does not claim this platform.

For macOS: The macOS platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

N/A, the TOE does not claim this platform.

2.2.2 Access to Platform Resources (FDP_DEC_EXT.1)

2.2.2.1 TSS Activities

FDP_DEC_EXT.1.1, FDP_DEC_EXT.1.2
None defined.

2.2.2.2 Guidance Activities

FDP_DEC_EXT.1.1
The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

[ST] indicates that the only hardware resource used by the TOE is network connectivity. As this is an implicit requirement for using a VPN client, the evaluator determined that it was not necessary for the guidance to explicitly list the hardware resources required for the TOE to function.

FDP_DEC_EXT.1.2
The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

[ST] states that no sensitive information repositories are accessed; therefore, no guidance is necessary for this.

2.2.2.3 Test Activities

Modified by TD0434 and TD0515.

FDP_DEC_EXT.1.1(1)
For Android: The evaluator shall verify that each <uses-permission>entry in the AndroidManifest.xml file for access to a hardware resource is reflected in the selection.

The evaluator verified that the AndroidManifest only provided references to certificates and network status information.

FDP_DEC_EXT.1.1(1)
For Windows: For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION, ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:

- http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

Per section 2.2.2.2 above, the only hardware resource required by the TOE is network connectivity. The application software and its documentation do not explicitly reference this but this is considered to be appropriate since the entire purpose of the product is to access network resources.

FDP_DEC_EXT.1.1(1)
For iOS: The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.

N/A, the TOE does not claim this platform.

FDP_DEC_EXT.1.1(1)
For Linux: The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

The evaluator verified the access restrictions are present in the documentation.

FDP_DEC_EXT.1.1(1)
For Solaris: The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

N/A, the TOE does not claim this platform.

FDP_DEC_EXT.1.1(1)
For macOS: The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

N/A, the TOE does not claim this platform.

FDP_DEC_EXT.1.2(1)

For Android: The evaluator shall verify that each <uses-permission>entry in the AndroidManifest.xml file for access to a sensitive information repository is reflected in the selection.

The evaluator verified that the AndroidManifest only provided references to certificates and network status information.

FDP_DEC_EXT.1.2(1)

For Windows: For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

- http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

The evaluator verified the access restrictions are present in the documentation.

FDP_DEC_EXT.1.2(1)

For iOS: The evaluator shall verify that either the application software or its documentation provides a list of the sensitive information repositories it accesses.

N/A, the TOE does not claim this platform.

FDP_DEC_EXT.1.2(1)

For Linux: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

The evaluator verified the access restrictions are present in the documentation.

FDP_DEC_EXT.1.2(1)

For Solaris: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

N/A, the TOE does not claim this platform.

FDP_DEC_EXT.1.2(1)

For macOS: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

N/A, the TOE does not claim this platform.

2.2.3 Network Communications (FDP_NET_EXT.1)

2.2.3.1 TSS Activities

None defined.

2.2.3.2 Guidance Activities

None defined.

2.2.3.3 Test Activities

The evaluator shall perform the following tests:

Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.

The evaluator observed the traffic from the TOE while a connection was present. The evaluator verified that all communication from the TOE are either documented or user-initiated.

Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

The evaluator performed a NMAP scan of the device running the TOE and verified that the TOE did not open any extra ports.

For Android: If "no network communication" is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a uses-permission or usespermission- sdk-23 tag containing android:name="android.permission.INTERNET". In this case, it is not necessary to perform the above Tests 1 and 2, as the platform will not allow the application to perform any network communication.

N/A, the specified selection is not made.

2.2.4 Full Residual Information Protection (FDP_RIP.2)

2.2.4.1 TSS Activities

Requirement met by the platform

The evaluator shall examine the TSS to verify that it describes (for each supported platform) the extent to which the client processes network packets and addresses the FDP_RIP.2 requirement.

N/A, this selection is not chosen.

Requirement met by the TOE

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

[ST] section 6.2.4 states that all packet data will be used to overwrite any previous buffer data, and if the packet is smaller than the allocated size of the buffer, the buffer is filled out with zeroes so that any residual data in the buffer is removed. This is the same for all claimed platforms.

2.2.4.2 Guidance Activities

There are no AGD EAs for this requirement.

2.2.4.3 Test Activities

There are no test EAs for this requirement.

2.3 Identification and Authentication (FIA)

2.3.1 Pre-Shared Key Composition (FIA_PSK_EXT.1)

2.3.1.1 TSS Activities

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.

[ST] does not claim the generation of bit-based pre-shared keys.

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

[ST] section 6.3.1 states that pre-shared keys between 1 and 256 characters are supported (which includes 22 characters), and that conditioning on the key is done by ASCII-binary conversion, consistent with the claims made in the SFR.

2.3.1.2 Guidance Activities

If the TOE supports bit-based pre-shared keys, 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 TOE supports bit-based pre-shared keys; per [ST] the TOE accepts bit-based pre-shared keys and does not generate them. The “Authentication Methods Supported in VIA” section of [User] has a chapter called “Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features” – under the Pre-Shared Keys heading, it describes how to enter pre-shared keys on the VPN gateway and deploy the key to registered clients.

The evaluator shall check that any management functions related to pre-shared keys that are performed by the TOE are specified in the operational guidance.

The only management function related to pre-shared keys is the act of entering the pre-shared key; this is described above.

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

[CCECG] includes guidance on pre-shared key length, allowed characters, and best practices. Specifically, section 2.2.6, where configuration of authentication mechanisms is described, states that a pre-shared key should be at least 22 characters and should include at least one uppercase letter, one lowercase letter, one number, and one special character. This section also identifies the special characters that are supported.

2.3.1.3 Test Activities

The evaluator shall perform the following tests:

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

The evaluator observed that this is controlled fully by the Mobility Controller (Gateway) that controls the TOE. The evaluator observed that the Gateway was capable of setting the pre-shared key value of the specified length and characters. The evaluator verified that the TOE is capable of completing the VPN connection with the use of a pre-shared key, which is pushed down to the TOE through the configurations made by the gateway.

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 invalid lengths that are below the minimum length, above the maximum length, null length, empty length, or zero length. The minimum and maximum length tests should be successful, and the invalid lengths must be rejected by the TOE.

The evaluator observed that the gateway is capable of configuring pre-shared keys of various lengths.

Test 3 [conditional]: If the TOE supports but 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 per the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

The evaluator observed that it is possible to configure PSK authentication on the gateway using a bit-based pre-shared key which is then communicated to the TOE and used in a VPN connection in the same manner as FIA_PSK_EXT.1 Test 1 above.

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.

N/A, the TOE requires the input of a key and does not perform generation of a pre-shared key.

2.3.2 X.509 Certificate Validation (FIA_X509_EXT.1)

2.3.2.1 TSS Activities

FIA_X509_EXT.1.1
The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

[ST] section 6.3.2 states that certificate validity checking is done when the server certificate is presented to the TOE and after the reference identifier is validated. The path validation algorithm attempts to construct a certificate path from the server certificate through any intermediate CAs up to the root CA. Once the path is validated, the TOE will check the validity of each certificate, followed by the revocation status. The TOE's behavior in response to revocation status is configured by the VPN gateway.

FIA_X509_EXT.1.2
None defined.

2.3.2.2 Guidance Activities

FIA_X509_EXT.1.1, FIA_X509_EXT.1.2
None defined.

2.3.2.3 Test Activities

Modified by TD0601.

FIA_X509_EXT.1.1
The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

FIA_X509_EXT.1.1
Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
- by omitting the basicConstraints field in one of the issuing certificates,
- by setting the basicConstraints field in an issuing certificate to have CA=False,
- by omitting the CA signing bit of the key usage field in an issuing certificate, and
- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

The evaluator attempted to perform a connection where the gateway was configured to present a certificate that met each of the specified cases. The evaluator verified that the TOE rejected the connection and did not complete an IPSEC channel when the gateway presented each of these certificates in turn.

FIA_X509_EXT.1.1

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

The evaluator observed that the gateway would not allow an administrator to configure a certificate that has expired. The evaluator observed that the gateway allowed a certificate that would expire in the near future (within the next 2 hours) to be configured for presentation to the TOE. The evaluator then observed that once the certificate expired the TOE rejected connections to the gateway until the certificate presented by the gateway had been changed to a valid certificate.

FIA_X509_EXT.1.1

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-"conditional on whether CRL, OCSP, OCSP Stapling or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

The evaluator shall test revocation of the node certificate.

The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP stapling per RFC 6066 is the only supported revocation method, this test is omitted.

The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

The evaluator observed that the TOE is capable of performing OCSP validation against certificates presented. The evaluator observed that the TOE is capable of validating end-entity and Intermediate-CA certificates are valid and if either the end-entity or Intermediate-CA certificate report as revoked that the TOE will not complete a connection.

FIA_X509_EXT.1.1

Test 4: If any OCSP option is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.

The evaluator observed that the TOE rejected the OCSP response if the OCSP responder does not possess the OCSP Signing purpose and did not allow the connection to the gateway to be completed.

FIA_X509_EXT.1.1

Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

The evaluator observed that the gateway did not permit such a certificate to be uploaded. The TOE is only capable of communicating to the gateway. When the TOE's operational environment is configured

as expected by the operational guidance, it is not expected that the TOE would be presented with such a certificate.

FIA_X509_EXT.1.1

Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator observed that the gateway did not permit such a certificate to be uploaded. The TOE is only capable of communicating to the gateway. When the TOE's operational environment is configured as expected by the operational guidance, it is not expected that the TOE would be presented with such a certificate.

FIA_X509_EXT.1.1

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator uploaded a certificate which has a modification in the public key to the gateway. The evaluator then configured the gateway to present the certificate. The evaluator observed that the TOE rejected connections to the gateway when this certificate was presented.

FIA_X509_EXT.1.1

Test 8a: (Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

The evaluator observed that the TOE was able to complete connections to the gateway when the gateway presented a certificate using ECDSA certificate chains where the chain used named-curve parameters.

FIA_X509_EXT.1.1

Test 8b: (Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

The evaluator observed that the TOE rejected connections to the gateway when the gateway presented certificates where the intermediate CA used the explicit curve parameters as a ECDSA certificate chain.

Modified by TD0495.

FIA_X509_EXT.1.2

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be

tested, two Intermediate CAs, and the selfsigned Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

FIA_X509_EXT.1.2

Test 1: The evaluator shall ensure that the certificate of at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.

The evaluator verified that the TOE was capable of validating that the CA's in the presented certificate chain that lack the basicConstraints extension are rejected when validating the gateway certificate.

FIA_X509_EXT.1.2

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

The evaluator verified that the TOE was capable of validating that the CA's in the presented certificate chain where the basicConstraint extension is set to FALSE are rejected when validating the gateway certificate.

2.3.3 X.509 Certificate Authentication (FIA_X509_EXT.2)

2.3.3.1 TSS Activities

The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

[ST] section 6.3.3 notes that the TOE chooses the certificate to use based on the assigned VPN profile, which determines the connection settings, including the connection certificate. When the validity of a certificate cannot be determined because an OCSP responder is unavailable, the TOE's behavior is to reject the certificate.

The "Configuring Pre-shared Keys and IKEv1/IKEv2 Authentication Features" chapter in the "Authentication Methods Supported in VIA" section of [User] has a heading called "OCSP"; under this it states that there is a setting that configures whether the connection is established if the OCSP status is unknown.

2.3.3.2 Guidance Activities

None defined.

2.3.3.3 Test Activities

The evaluator shall perform the following test for each trusted channel:

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

The evaluator observed that the TOE is able to query the OCSP revocation status from the remote OCSP responder in the test environment. The evaluator observed that when the status could be received, verified, and was valid the connection could be completed. The evaluator observed that if the OCSP responder was turned off the connection could not be completed.

Test 2: The evaluator shall demonstrate that an invalid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity cannot be accepted.

The evaluator observed that if an invalid certificate which presented revocation information was presented to the TOE the TOE rejects the connection after querying the OCSP status.

2.4 Security Management (FMT)

2.4.1 Secure by Default Configuration (FMT_CFG_EXT.1)

2.4.1.1 TSS Activities

FMT_CFG_EXT.1.1
The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials.

[ST] section 6.4.1 states that the TOE does not have any default credentials. Credentials are specified by the user when initiating a connection attempt.

FMT_CFG_EXT.1.2
None defined.

2.4.1.2 Guidance Activities

FMT_CFG_EXT.1.1, FMT_CFG_EXT.1.2
None defined.

2.4.1.3 Test Activities

If the application uses any default credentials the evaluator shall run the following tests.

FMT_CFG_EXT.1.1

Test 1: The evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.

N/A, as the TOE does not use any default credentials.

FMT_CFG_EXT.1.1

Test 2: The evaluator shall attempt to clear all credentials and verify that only the minimal application functionality required to set new credentials is available.

N/A, as the TOE does not use any default credentials.

FMT_CFG_EXT.1.1

Test 3: The evaluator shall run the application, establish new credentials and verify that the original default credentials no longer provide access to the application.

N/A, as the TOE does not use any default credentials.

FMT_CFG_EXT.1.2 The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

FMT_CFG_EXT.1.2 For Android: The evaluator shall run ls -alR|grep -E '^.....w.' inside the application's data directories to ensure that all files are not world-writable. The command should not print any files. The evaluator shall also verify that no sensitive data is written to external storage which could be read/modified by any application containing the READ_EXTERNAL_STORAGE and/or WRITE_EXTERNAL_STORAGE permissions.

The evaluator preformed the specified command and observed that no files were written in the applications data directories that are world-writable.

FMT_CFG_EXT.1.2 For Windows: The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.

The evaluator used the PowerShell Get-Acl command, which is equivalent to the capabilities of icalcs, to verify that all files in the TOE's installation directory are protected from regular user modification.

FMT_CFG_EXT.1.2 For iOS: The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.

N/A, the TOE does not claim this platform.

Modified by TD0519.

FMT_CFG_EXT.1.2 For Linux: The evaluator shall run the command find-L . -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

The evaluator verified that no world-writeable files were found in the application data directory.

FMT_CFG_EXT.1.2 For Solaris: The evaluator shall run the command find . \(-perm -002 \) inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

N/A, the TOE does not claim this platform.

FMT_CFG_EXT.1.2 For macOS: The evaluator shall run the command find . -perm +002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

N/A, the TOE does not claim this platform.

2.4.2 Supported Configuration Mechanism (FMT_MEC_EXT.1)

Modified by TD0437.

2.4.2.1 TSS Activities

The evaluator shall review the TSS to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform or implemented by the application in accordance with the PP-Module for File Encryption. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

[ST] section 6.4.2 states that the TOE's configuration options are stored on the TOE platform. The Windows platform version stores settings in the registry, the Android version of the TOE stores settings in /data/data/<vendor-specific directory> using SharedPreferences, and the Linux version of the TOE stores settings in the user profile and home directory. The security-relevant configuration settings are the URL(s) of the VPN gateway and connection profile(s).

Conditional: If "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption" is selected, the evaluator shall ensure that the TSS identifies those options, as well as indicates where the encrypted representation of these options is stored.

This selection is not chosen so the evaluation activity is not applicable.

2.4.2.2 Guidance Activities

None defined.

2.4.2.3 Test Activities

If "invoke the mechanisms recommended by the platform vendor for storing and setting configuration options" is chosen, the method of testing varies per platform as follows:

For Android: The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one XML file at location /data/data/package/shared_prefs/ reflects the changes made to the configuration to verify that the application used SharedPreferences and/or PreferenceActivity classes for storing configuration data, where package is the Java package of the application.

The evaluator observed the application preferences xml to observe the changes were made in the correct location.

Modified by TD0465 and 0543.

For Windows: The evaluator shall determine and verify that Windows Universal Applications use either the Windows.Storage namespace, Windows.UI.ApplicationSettings namespace or the IsolatedStorageSettings namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the Windows Registry or C:\ProgramData\ directory.

The TOE is a Classic Desktop application. The evaluator utilized ProcMon to verify that the TOE stores the configuration changes made by the TOE are stored correctly.

For iOS: The evaluator shall verify that the app uses the user defaults system or key-value store for storing all settings.

N/A, the TOE does not claim this platform.

For Linux: The evaluator shall run the application while monitoring it with the utility strace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that strace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration), in the user's home directory (for user-specific configuration), or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).

The evaluator utilized strace to verify that the TOE stores configurations changes made by the TOE are stored correctly.

For Solaris: The evaluator shall run the application while monitoring it with the utility dtrace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that dtrace logs corresponding changes to configuration files that reside in /etc (for system-specific configuration) or in the user's home directory(for user-specific configuration).

N/A, the TOE does not claim this platform.

For macOS: The evaluator shall verify that the application stores and retrieves settings using the NSUserDefaults class.

N/A, the TOE does not claim this platform.

If " implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption" is selected, for all configuration options listed in the TSS as being

stored and protected using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

N/A the specified module is not claimed.

2.4.3 Specification of Management Functions (FMT_SMF.1)

2.4.3.1 TSS Activities

None defined.

2.4.3.2 Guidance Activities

The evaluator shall verify 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.

[ST] does not claim any management functions beyond those specified in FMT_SMF.1/VPN, which is addressed in section 2.4.4 below.

2.4.3.3 Test Activities

The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

The TOE only supports the VPN functionality and thus this was fully tested by performing the testing required.

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

2.4.4.1 TSS Activities

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

[ST] section 6.4.4 states that client credentials are used for IPsec connections. It is understood by the evaluator that the TOE is a software application that is launched from an existing OS session so there is no “logging in” to the TOE; instead, the TOE is used to establish a connection to an external VPN gateway, which requires some form of authentication to allow the IPsec connection to be established.

2.4.4.2 Guidance Activities

The evaluator shall check to make sure that every management function mandated in the ST for this requirement is described in the operational guidance and that the description contains the information required to perform the management duties associated with each management function.

[ST] defines the following management functions, for which guidance can be found in the following locations:

- Specify VPN gateways to use for connections: the “VIA Client for Windows”, “VIA Client for Linux”, and “VIA Client for Android” sections of [User] each have chapters called “Configuring VIA Settings” – instructions for selecting the server can be found under the “Changing the Server” dropdown.
- Specify client credentials to be used for connections: the “VIA Client for Windows”, “VIA Client for Linux”, and “VIA Client for Android” sections of [User] each have chapters called “Connecting and Disconnecting VIA” – instructions for selecting certificate credentials can be found under the “Certificate-Based Authentication” and “Certificate-Based Authentication with Extended Authentication (XAUTH)” headers.

2.4.4.3 Test Activities

The evaluator shall test the TOE’s ability to provide the management functions by configuring the TOE according to the operational guidance and testing each management activity listed in the ST.

The evaluator verified that the TOE is capable of configuring the VPN functionality by connecting to the Gateway and the evaluator observed that the TOE could enable/disable the connection status.

The evaluator shall ensure that all management functions claimed in the ST can be performed by completing activities described in the AGD. Note that this may be performed in the course of completing other testing.

The following summary of the test activity performed by the evaluator references the management functions claimed in FMT_SMF.1.1/VPN with the claimed functions in **bold**.

The evaluator configured the TOE to connect to a specified VPN gateway **(Specify VPN gateways to use for connections)** and entered the credentials to be used for the connection **(Specify client credentials to be used for connections)**.

2.5 Privacy (FPR)

2.5.1 User Consent for Transmission of Personally Identifiable Information (FPR_ANO_EXT.1)

2.5.1.1 TSS Activities

The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

[ST] section 6.5.1 states that the TOE does not transmit personally identifiable information over a network. Based on the functionality of the TOE, it is clear that this is accurate; the TOE itself only transmits information needed to establish an IPsec connection.

2.5.1.2 Guidance Activities

None defined.

2.5.1.3 Test Activities

If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

"Require user approval before executing…" is not selected in [ST] so this test activity is N/A.

2.6 Protection of the TSF (FPT)

2.6.1 Anti-Exploitation Capabilities (FPT_AEX_EXT.1)

2.6.1.1 TSS Activities

FPT_AEX_EXT.1.1 The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

[ST] section 6.6.1 states that the TOE uses /DYNAMICBASE to support ASLR for Windows, and -fpic for Android and Linux.

FPT_AEX_EXT.1.2, FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4, FPT_AEX_EXT.1.5 None defined.

2.6.1.2 Guidance Activities

FPT_AEX_EXT.1.1, FPT_AEX_EXT.1.2, FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4, FPT_AEX_EXT.1.5 None defined.

2.6.1.3 Test Activities

Modified by TD0544.

FPT_AEX_EXT.1.1 The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.
FPT_AEX_EXT.1.1 **For Android:** The evaluator shall run the same application on two different Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool. Connect via ADB and inspect /proc/PID/maps. Ensure the two different instances share no memory mappings made by the application at the same location.

The evaluator ran the application on two different android devices and observed the pid maps on each and verified that the TOE did not share any memory maps between both devices.

FPT_AEX_EXT.1.1

For Windows: The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

The evaluator verified that the windows application had the ALSR enablement flags present.

FPT_AEX_EXT.1.1

For iOS: The evaluator shall perform a static analysis to search for any mmap calls (or API calls that call mmap), and ensure that no arguments are provided that request a mapping at a fixed address.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.1

For Linux: The evaluator shall run the same application on two different Linux systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

The evaluator observed that the TOE did not use a static memory mapping by running pmap on the TOE at two different times and verifying that the values are the different.

FPT_AEX_EXT.1.1

For Solaris: The evaluator shall run the same application on two different Solaris systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.1

For macOS: The evaluator shall run the same application on two different Mac systems. The evaluator shall then compare their memory maps using vmmap PID to ensure the two different instances share no mapping locations.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.2

The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

FPT_AEX_EXT.1.2

For Android: The evaluator shall perform static analysis on the application to verify that

- mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked.

The evaluator observed the static analysis results that were provided by the vendor and found that mmap and mprotect were not present in the results.

FPT_AEX_EXT.1.2

For Windows: The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

The evaluator observed that the application passed the NXCheck from binscope.

FPT_AEX_EXT.1.2

For iOS: The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.2

For Linux: The evaluator shall perform static analysis on the application to verify that both

- mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked with the PROT_EXEC permission.

The evaluator observed the static analysis results that were provided by the vendor and found that mmap and mprotect were not present in the results.

FPT_AEX_EXT.1.2

For Solaris: The evaluator shall perform static analysis on the application to verify that

both

- mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked with the PROT_EXEC permission.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.2

For macOS: The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.3

The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

FPT_AEX_EXT.1.3

For Android: Applications running on Android cannot disable Android security features, therefore this requirement is met and no evaluation activity is required.

Per the Evaluation Activity, this test is met implicitly and no testing is needed.

FPT_AEX_EXT.1.3

For Windows: If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled;

Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threatprotection/ windows-defender-exploit-guard/customize-exploit-protection. If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

The evaluator observed that the TOE was capable of running on a windows device that has CFG, Bottom up ASLR, EAF, IAF, and DEP present.

FPT_AEX_EXT.1.3

For iOS: Applications running on iOS cannot disable security features, therefore this requirement is met and no evaluation activity is required.

N/A, the TOE does not claim this platform.

Modified by TD0435.

FPT_AEX_EXT.1.3

For Linux: The evaluator shall ensure that the application can successfully run on a system with either SELinux or AppArmor enabled and in enforce mode.

The evaluator observed that the TOE was capable of running on a Ubuntu system with AppArmor enabled and running.

FPT_AEX_EXT.1.3

For Solaris: The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.3

For macOS: The evaluator shall ensure that the application can successfully run on macOS without disabling any security features.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.4

The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

FPT_AEX_EXT.1.4

For Android: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored under /data/data/package/ where package is the Java package of the application.

The evaluator observed that the TOE stored no executable files in the specified location.

Modified by TD0445.

FPT_AEX_EXT.1.4

For Windows: For Windows Universal Applications the evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

The evaluator observed that the TOE stored no executable files in the same directory as user-modifiable files.

FPT_AEX_EXT.1.4

For iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.4

For Linux: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

The evaluator observed that the TOE stored no executable files in the same directory as user-modifiable files.

FPT_AEX_EXT.1.4

For Solaris: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.4

For macOS: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.5

The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

FPT_AEX_EXT.1.5

For Windows: Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS

and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

The evaluator observed that the TOE passed the binscope check for GSCheck.

FPT_AEX_EXT.1.5
For PE , the evaluator will disassemble each and ensure the following sequence appears:
mov rcx, QWORD PTR [rsp+ (...)]
xor rcx, (...)
call (...)
.

N/A, the TOE does not claim this platform.

FPT_AEX_EXT.1.5
For ELF executables, the evaluator will ensure that each contains references to the symbol **__stack_chk_fail**.

The evaluator verified that the linux TOE contained a reference to the __stack_chk_fail symbol.

FPT_AEX_EXT.1.5
Tools such as Canary Detector may help automate these activities.

2.6.2 Use of Supported Services and APIs (FPT_API_EXT.1)

2.6.2.1 TSS Activities

The evaluator shall verify that the TSS lists the platform APIs used in the application.

[ST] section 6.6.2 references the appendices of the ST where platform APIs are listed for each of the TOE's supported platform versions (Windows, Linux, Android).

2.6.2.2 Guidance Activities

None defined.

2.6.2.3 Test Activities

The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

The evaluator verified that the API's listed in the ST are publicly documented.

2.6.3 Software Identification and Versions (FPT_IDV_EXT.1)

2.6.3.1 TSS Activities

If "other version information" is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology.

[ST] section 6.6.3 identifies the TOE's version methodology as <major>.<minor>.<patch>.<build>, where major/minor version is used for feature updates and patch/build version is used for security patches and bug fixes.

2.6.3.2 Guidance Activities

None defined.

2.6.3.3 Test Activities

The evaluator shall install the application, then check for the / existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

The evaluator observed that the TOE was capable of reporting it's own version. The TOE does not claim the use of SWID tags.

2.6.4 Use of Third Party Libraries (FPT_LIB_EXT.1)

2.6.4.1 TSS Activities

None defined.

2.6.4.2 Guidance Activities

None defined.

2.6.4.3 Test Activities

The evaluator shall install the application and survey its installation directory for dynamic libraries. The evaluator shall verify that libraries found to be packaged with or employed by the application are limited to those in the assignment.

The evaluator installed the TOE and observed the installation directory for dynamic libraries. The evaluator verified that the libraries present are documented.

2.6.5 TSF Self-Test (VPN Client) (FPT_TST_EXT.1/VPN)

Note this SFR is iterated by the ST author because there are some differences between platform versions in how the requirements are implemented. All platform versions are described collectively below.

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

2.6.5.1 TSS Activities

The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF

is operating correctly. If some of the tests are performed by the TOE platform, the evaluator shall check the TSS to ensure that those tests are identified, and that the ST for each platform contains a description of those tests. Note that the tests that are required by this component are those that support security functionality in the VPN Client PP-Module, which may not correspond to the set of all self-tests contained in the platform STs.

[ST] section 6.6.5 states that all platform versions of the TOE perform a series of cryptographic self-tests at power-on to ensure that the built-in cryptographic library is functioning correctly. The tests (cryptographic algorithms, pairwise consistency, cryptographic known-answer) are understood to be typical mechanisms for checking that cryptographic algorithms are implemented in a mathematically correct manner.

This section also outlines the software integrity tests that are performed. Specifically, the Windows and Linux platform versions of the TOE perform a software integrity test on the cryptographic module, and the Android version of the TOE relies on the OS platform to perform an integrity test on the entire application. This section argues that the security functionality of the TOE rests entirely within the cryptographic module (i.e. the rest of the application itself is essentially just a wrapper that allows for the user to specify what module functions are invoked) and so the integrity check on the module itself is sufficient to ensure that the IPsec VPN client functionality cannot be invoked in an insecure manner.

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

The cryptographic functionality and software integrity tests specified in [ST] section 6.6.5 are sufficient to ensure that security-relevant tampering of the product has not occurred because any alteration of the product's cryptographic implementation would be detected, which means there is no situation in which an undetected modification of the underlying IPsec functionality would occur.

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

[ST] section 6.6.5 states that the TOE will fail to start in the event of a failed self-test. If the self-test is successful, it can be assumed that the TOE starts as normal, and that the absence of any obvious failure indicates a successful check. This section also notes that an error message will be placed in the local debug log file and identifies the messages that will be placed there based on the platform.

2.6.5.2 Guidance Activities

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

In addition to the materials referenced above in [ST], section 2.4 of [CCECG] also includes the successful and failed self-test log messages as well as troubleshooting steps that can be followed if a self-test failure is detected.

2.6.5.3 Test Activities

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.

The evaluator verified that the cryptographic library was capable of performing the integrity check on startup.

Test 2: The evaluator modifies the TSF executable, performs the integrity check on the modified TSF executable and verifies that the check fails.

The evaluator verified that the cryptographic library failed to load when the integrity check failed, in this situation the TOE also does not successfully start.

2.6.6 Trusted Update (FPT_TUD_EXT.1)

Note this SFR is iterated by the ST author because there are some differences between platform versions in how the requirements are implemented. All platform versions are described collectively below.

FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, FPT_TUD_EXT.1.3 None defined.

Modified by TD0561.

FPT_TUD_EXT.1.4 The evaluator shall verify that the TSS identifies how updates to the application are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

[ST] section 6.6.6 states that the Android version of the TOE uses the Google Play store, which uses a Google signature to guarantee the authenticity of the application. This is also where candidate updates are obtained for this version. For the Windows and Linux platforms, [ST] section 6.6.6 states that candidate updates are obtained through the Mobility Controller (VPN gateway) or Aruba Support Portal and that the update is signed by the vendor regardless of its origin.

FPT_TUD_EXT.1.5 The evaluator shall verify that the TSS identifies how the application is distributed. If "with the platform" is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If "as an additional package" is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

It is clear from [ST] that the TOE is third-party software that is not an innate part of a desktop or mobile operating system (e.g. because the TOE does not include a GPOS or MDF PP Base-PP claim); testing for FPT_TUD_EXT.2 was therefore performed.

2.6.6.1 Guidance Activities

FPT_TUD_EXT.1.1
The evaluator shall check to ensure the guidance includes a description of how updates are performed.

Section 2.1 of [CCECG] states that TOE updates are obtained from the Google Play store for Android, and from the Aruba website for Windows/Linux. Section 2.5 of [CCECG] also states that updates can alternatively be configured for Windows/Linux to be obtained from the Mobility Controller (VPN gateway). If the update is successful, the updated version can be seen on the TOE.

FPT_TUD_EXT.1.2
The evaluator shall verify guidance includes a description of how to query the current version of the application.

The "VIA Client for Windows", "VIA Client for Linux", and "VIA Client for Android" sections of [User] each have chapters called "Configuring VIA Settings" – the "About" section states that the About tab displays the current running version of the TOE. This section also states how a user is made aware of a failed update.

FPT_TUD_EXT.1.3, FPT_TUD_EXT.1.4, FPT_TUD_EXT.1.5
None defined.

N/A

2.6.6.2 Test Activities

FPT_TUD_EXT.1.1(1)
The evaluator shall check for an update using procedures described in either the application documentation or the platform documentation and verify that the application does not issue an error. If it is updated or if it reports that no update is available this requirement is considered to be met.

The evaluator verified that the TOE is capable of checking for updates according to the method specified for each platform. The evaluator verified that there were no updates available.

FPT_TUD_EXT.1.2(1)
The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

The evaluator observed that the TOE is capable of reporting its own current version. The evaluator verified that this version matches the specified version.

Modified by TD0548.

FPT_TUD_EXT.1.3(1)
For iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

N/A, the TOE does not claim this platform.

FPT_TUD_EXT.1.3(1)
For all other platforms: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

The evaluator observed all of the TOE's executable files and saved off a hash of each of the files. the evaluator then executed the TOE and then captured the hash of each of the executable files again. The evaluator verified that the hash values were the same between both captures.

FPT_TUD_EXT.1.4(1), FPT_TUD_EXT.1.5(1)
None defined.

2.6.7 Integrity for Installation and Update (FPT_TUD_EXT.2)

Modified by TD0561.

2.6.7.1 TSS Activities

FPT_TUD_EXT.2.1, FPT_TUD_EXT.2.2
None defined.

N/A

FPT_TUD_EXT.2.3
The evaluator shall verify that the TSS identifies how the application installation package is signed by an authorized source. The definition of an authorized source must be contained in the TSS.

[ST] section 6.6.6 states the product vendor (Aruba) is the authorized source for the signature for Windows and Linux software updates, and that Google is the authorized source for the signature for Android updates since those updates originate from the Google Play Store.

2.6.7.2 Guidance Activities

FPT_TUD_EXT.2.1, FPT_TUD_EXT.2.2, FPT_TUD_EXT.2.3
None defined.

2.6.7.3 Test Activities

FPT_TUD_EXT.2.1
The evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:

FPT_TUD_EXT.2.1
For Android: The evaluator shall ensure that the application is packaged in the Android application package (APK) format.

The evaluator verified that the TOE was packaged as an APK and was available in the play store.

FPT_TUD_EXT.2.1
For Windows: The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See https://msdn.microsoft.com/enus/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.

The evaluator verified that the TOE was packaged as a MSI file.

FPT_TUD_EXT.2.1
For iOS: The evaluator shall ensure that the application is packaged in the IPA format.

N/A, the TOE does not claim this platform.

FPT_TUD_EXT.2.1
For Linux: The evaluator shall ensure that the application is packaged in the format of the package management infrastructure of the chosen distribution. For example, applications running on Red Hat and Red Hat derivatives shall be packaged in RPM format. Applications running on Debian and Debian derivatives shall be packaged in DEB format.

The evaluator observed that the TOE was packaged as a .deb file for DEB based operating systems.

FPT_TUD_EXT.2.1
For Solaris: The evaluator shall ensure that the application is packaged in the PKG format.

N/A, the TOE does not claim this platform.

FPT_TUD_EXT.2.1
For macOS: The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format.

N/A, the TOE does not claim this platform.

FPT_TUD_EXT.2.2
For Android: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Per the evaluation activity the evaluator considers this requirement to be met.

FPT_TUD_EXT.2.2
For iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

N/A, the TOE does not claim this platform.

FPT_TUD_EXT.2.2
For All Other Platforms: The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application.

Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

The evaluator captured all of the files on the files system prior to installing the TOE. The evaluator then installed and ran the TOE. The evaluator then captured the files on the file system after uninstalling the TOE and verified that the TOE correctly uninstall's itself.

FPT_TUD_EXT.2.3
None defined.

2.7 Trusted Path/Channels (FTP)

2.7.1 Protection of Data in Transit (FTP_DIT_EXT.1)

2.7.1.1 TSS Activities

For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

The TOE does not claim platform-provided functionality for this SFR so this evaluation activity is not applicable.

From [MOD VPNC] per TD0622:
For IPsec, refer to the Evaluation Activity for FCS_IPSEC_EXT.1 in Section 2.5.1.2. If other protocols are selected for FTP_DIT_EXT.1, refer to the Evaluation Activity for FTP_DIT_EXT.1 in the App PP.

IPsec materials are referenced in section 2.1.12 above.

2.7.1.2 Guidance Activities

Modified by TD0601 and TD0622

None defined.

From [MOD VPNC] per TD0622:
For IPsec, refer to the Evaluation Activity for FCS_IPSEC_EXT.1 in Section 2.5.1.2. If other protocols are selected for FTP_DIT_EXT.1, refer to the Evaluation Activity for FTP_DIT_EXT.1 in the App PP.

IPsec materials are referenced in section 2.1.12 above.

2.7.1.3 Test Activities

Modified by TD0601 and TD0622.

From [MOD VPNC] per TD0622:
For IPsec, refer to the Evaluation Activity for FCS_IPSEC_EXT.1 in Section 2.5.1.2. If other protocols are selected for FTP_DIT_EXT.1, refer to the Evaluation Activity for FTP_DIT_EXT.1 in the App PP.

From [APP PP]

The evaluator shall perform the following tests.

Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS, DTLS, SSH, or IPsec in accordance with the selection in the ST.

No protocols other than IPsec are selected. Per TD0622, evaluation activities specified in that TD are sufficient to cover IPsec, and the FTP_DIT_EXT.1 testing from [APP PP] is only referenced if other protocols are selected. Since no other protocols are selected, this evaluation activity is not applicable.

Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

No protocols other than IPsec are selected. Per TD0622, evaluation activities specified in that TD are sufficient to cover IPsec, and the FTP_DIT_EXT.1 testing from [APP PP] is only referenced if other protocols are selected. Since no other protocols are selected, this evaluation activity is not applicable.

Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

No protocols other than IPsec are selected. Per TD0622, evaluation activities specified in that TD are sufficient to cover IPsec, and the FTP_DIT_EXT.1 testing from [APP PP] is only referenced if other protocols are selected. Since no other protocols are selected, this evaluation activity is not applicable.

For Android: If "not transmit any data" is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a uses-permission or uses-permission-sdk-23 tag containing android:name="android.permission.INTERNET". In this case, it is not necessary to perform the above Tests 1, 2, or 3, as the platform will not allow the application to perform any network communication.

N/A this selection is not made.

For iOS: If "encrypt all transmitted data" is selected, the evaluator shall ensure that the application's Info.plist file does not contain the NSAllowsArbitraryLoads or NSExceptionAllowsInsecureHTTPLoads keys, as these keys disable iOS's Application Transport Security feature.

N/A, the TOE does not claim this platform.

3 Security Assurance Requirement Assurance Activities

3.1 Security Target (ASE)

As per ASE activities defined in [CEM].

CEM ASE evaluation activities are addressed in the proprietary Evaluation Technical Report (ETR)

3.2 Development (ADV)

3.2.1 Basic Functional Specification (ADV_FSP.1)

3.2.1.1 Assurance Activity

There are no specific evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in 5.1 Security Functional Requirements, 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 evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

Through review of the ST and guidance documentation, it is clear that the TOE's external interfaces include the following:

- VPN interface: used to establish IPsec connectivity with Mobility Controller
- User interface: used to perform basic operational tasks such as selecting networks to connect to, connecting to/disconnecting from the network, and entering credential data (i.e. authenticate to the VPN gateway). All platform versions have a graphical interface, while the Windows and Linux versions also have a command line interface and the Android version has a management API that can be used to perform many of the same functions.

The TOE also relies on the underlying OS platform's HTTPS interface to acquire configuration data for VPN connections, but this is not implemented by Aruba VIA itself.

3.3 Guidance Documents (AGD)

3.3.1 Operational User Guidance (AGD_OPE.1)

3.3.1.1 Assurance Activity

Some of the contents of the operational guidance will be verified by the evaluation activities in 5.1 Security Functional Requirements and evaluation of the TOE according to the [CEM]. The following additional information is also required.
If cryptographic functions are provided by the TOE, 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.

[CCECG] section 1 states that the FIPS mode of operation must be enabled for the TOE to be in its evaluated configuration and that this is done through configuration on the Mobility Controller.

The documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform. The evaluator shall verify that this process includes the following steps:

- 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 digital signature.
- The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

[CCECG] states that TOE updates are obtained from the Google Play store for Android, and from a designated URL for Windows/Linux.

The "VIA Client for Windows", "VIA Client for Linux", and "VIA Client for Android" sections of [User] each have chapters called "Configuring VIA Settings" – the "About" section states that the About tab displays the current running version of the TOE. This section also states how a user is made aware of a failed update.

The introductory section of [CCECG] identifies the product functionality that is excluded from the evaluated configuration of the TOE as a warning to the administrator. Specifically, non-FIPS mode was not evaluated, and only the Windows, Linux, and Android platform versions were evaluated (i.e. macOS and iOS are outside the scope of the evaluation).

3.3.2 Preparative Procedures (AGD_PRE.1)

3.3.2.1 Assurance Activity

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

[User] includes chapters for "VIA Client for Windows", "VIA Client for Linux", and "VIA Client for Android" – in each of them, there is an "Installing VIA client for <platform name>" chapter that describes how to obtain the installer for the file. In all cases, the installation package is either a well-known package format for which no special instructions are required (Windows .msi, Linux .rpm), or the installation occurs automatically through the platform's own processes once the installer has been acquired (Android installation after acquisition from Google Play store).

In these three chapters, the text-based client-side management functionality for each platform version of the TOE is listed (CLI for Windows and Linux, Management API for Android). In general, this functionality generally relates to an alternative mechanism for specifying configuration options that are otherwise generally available to the user in the GUI. However, in the case of Windows, the "VIA for Windows Command-Line Interface" chapter also provides guidance on customizing the installation of the TOE, specifically in regards to running the installation command with parameters that allow for the following settings:

- Whether or not a desktop shortcut is created
- Whether or not the default installation directory is overwritten with an administrator-specified directory
- Whether the application is given a custom name when viewed in the Start menu
- Whether or not the application is configured to automatically launch on system start

3.4 Life-cycle Support (ALC)

3.4.1 Labeling of the TOE (ALC_CMC.1)

3.4.1.1 Assurance Activity

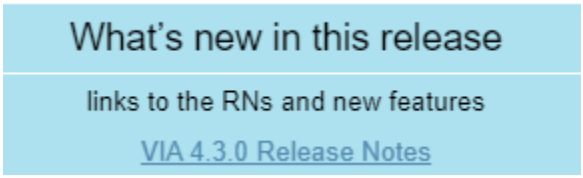
The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST.

[ST] identifies the TOE as Aruba Virtual Intranet Access (VIA) v4.3.

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.

[User] identifies the TOE as Virtual Intranet Access (VIA) 4.x.

Per FPT_IDV_EXT.1, it is understood that the .x is a minor release. The latest version of the TOE, listed on this page under "What's new in this release" identifies the latest version as 4.3.0, which is understood to be consistent with version 4.3 (the third number refers to a maintenance release number and is incremented for bug fixes or security patches that don't affect feature functionality).



The TOE itself also identifies its version, as shown below. The screenshot below is from the Android version of the TOE but it is representative of all TOE platform versions; each platform version has an About tab that presents the product version information in the same manner.

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.

The vendor maintains a web site advertising Aruba VPN products in general at https://www.arubanetworks.com/products/security/vpn-services/. This site references a product sheet about Aruba products that can be used to “Secure Your Workforce” (https://www.arubanetworks.com/assets/eo/AAG_Securing-Distributed-Workforce.pdf). This sheet includes a section heading for “Aruba VIA” that describes it as follows:

> VIA is a hybrid IPsec/SSL VPN client that automatically scans and selects the best, secure connection to terminate corporate-bound traffic. Unlike traditional VPNs which require dedicated hardware, Aruba integrates VPN services directly on existing Aruba secure infrastructure to simplify architecture and management. For military-grade security, VIA supports Suite B cryptography when used with the ArubaOS Advanced Cryptography (ACR) module. In this deployment model, mobile devices or desktop workstations can securely access networks that handle controlled unclassified, confidential and classified information.

The product's functionality is clear from this description, and it is consistent with the ST's definition of the product (except that SSL VPN capabilities are outside the scope of the evaluation because the TOE assumes that an Aruba Mobility Controller is configured as an IPsec VPN gateway).

The user guidance for VIA ([User]) includes a "What's new in this release" splash header which lists the most recent versions of the product; VIA 4.3.0 is listed as the latest product version (as of 2/2/22).

3.4.2 TOE CM Coverage (ALC_CMS.1)

3.4.2.1 Assurance Activity

The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the evaluation activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.
The evaluator shall ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform.
For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags).
The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled.
The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

The TOE's buffer overflow protections are enabled by default; no specific actions are required to enable this as it is an intrinsic property of the TOE's compilation process.

The evaluator verified that the TOE is interchangeably identified as Aruba Virtual Intranet Access, Aruba VIA, or simply VIA, that the product's user guidance identifies it as version 4.x (as the feature set will be the same from a user perspective across all minor versions), and that release notes identify the individual product releases within the 4.x version, most recently version 4.3.0.

3.4.3 Timely Security Updates (ALC_TSU_EXT.1)

3.4.3.1 Assurance Activity

The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates.

[ST] section 6.8 as a whole summarizes the timely security update process.

The evaluator shall verify that this description addresses the entire application.

[ST] section 6.8 does not stipulate that the timely security update process only applies to a subset of the TOE so it can be assumed that the description addresses the entire application.

The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description.

[ST] section 6.8 states that side channels for vulnerability disclosures such as CVEs are reviewed, such that the vendor's reporting mechanism is not the only method by which the vendor may become aware of a security flaw.

The evaluator shall also verify that each mechanism for deployment of security updates is described.

Per [ST] section 6.6.6, security updates are deployed as updated versions of the TOE software, where a maintenance release would be issued (e.g. an update to Aruba VIA 4.3.0 that addresses some security flaw could be labeled as version 4.3.1).

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment.

[ST] section 6.8 says that the timeliness of a response is dependent on the severity of a finding: high within 30 days, moderate within 90 days, and low within 180 days, but that findings have historically been addressed within 30 days regardless of severity.

The evaluator shall verify that this time is expressed in a number or range of days.

See above.

The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE.

[ST section 6.8 references https://www.arubanetworks.com/support-services/sirt/ as the mechanism for reporting security issues related to the TOE. This site describes the security reporting process and includes guidance for how to submit potential security issues.

The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

[ST] section 6.8 says that findings can be submitted directly to the vendor via email using a PGP key or by opening a ticket on the HTTPS support site.

3.5 Tests (ATE)

3.5.1 Independent Testing – Conformance (ATE_IND.1)

3.5.1.1 Assurance Activity

The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's evaluation activities.

The testing effort was documented in [Test].

While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator 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 platforms 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 effect; 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 evaluator is 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) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

[Test] documents the setup of each test with respect to the application SFRs claimed in [ST] and the associated Evaluation Activities.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (e.g SSH). 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.

[Test] describes the configuration of the cryptographic functions necessary to address the relevant test activities.

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.

The test report is documented as the completed version of [Test].

3.6 Vulnerability Assessment (AVA)

3.6.1 Vulnerability Survey (AVA_VAN.1)

Modified by TD0554.

3.6.1.1 Assurance Activity

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 find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses.

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. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

For Windows, Linux, macOS and Solaris: The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious.

The evaluation team performed a search of the following public vulnerability databases:

- National Vulnerability Database (https://nvd.nist.gov/)
- Aruba Security Advisories (https://www.arubanetworks.com/support-services/security-bulletins/)

Searches were performed several times, most recently on 08 August 2022, using the following search terms:

- aruba
- arubanetworks
- aruba via
- Virtual Intranet Client
- IPSec VPN Client

The evaluation team determined that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.

Note that Aruba disclosed a vulnerability in VIA 4.3 for Windows on July 26, 2022 that allows a man in the middle to intercept data exchanged during profile configuration, which may involve the exchange of security-relevant data. This issue was addressed in VIA 4.4 but Aruba additionally released a hotfix for VIA 4.3 on August 15, 2022 that addresses the issue on its own, without any of the feature updates included in version 4.4. Based on communications with the vendor, the specific nature of the fix does not affect any of the tested behavior as defined by the evaluation activities for the claimed SFRs. This hotfix was communicated to customers as an update on the Aruba Security Advisories page on August 19, 2022.

Per TD0554, the evaluator conducted virus scanning on Windows and Linux using McAfee Endpoint Security 10.7 using up-to-date definitions as of 6/23/2022 and observed that no potential malicious findings were present.