Microsoft Windows 10 Mobile and Microsoft Windows 10 with Lumia 950, Lumia 950 XL, Lumia 550, Lumia 635, and Surface Pro 4 Common Criteria Assurance Activities Report

Version 1.0

April 29, 2016

Prepared by:

Leidos Inc. (formerly Science Applications International Corporation)
https://www.leidos.com/commercialcyber/ate
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:

Microsoft Corporation

Corporate Headquarters
One Microsoft Way
Redmond, WA 98052-6399

The TOE Evaluation was Sponsored by:

Microsoft Corporation

Corporate Headquarters
One Microsoft Way
Redmond, WA 98052-6399

Evaluation Personnel:

Gary Grainger
Greg Beaver

Common Criteria Versions

- *Common Criteria for Information Technology Security Evaluation Part 1: Introduction,* Version 3.1, Revision 4, September 2012.
- *Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components,* Revision 4, September 2012.
- *Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components,* Revision 4, September 2012.

Common Evaluation Methodology Versions

- *Common Methodology for Information Technology Security Evaluation, Evaluation Methodology,* Version 3.1, Revision 4, September 2012.

Protection Profiles

1. *Protection Profile for Mobile Device Fundamentals,* Version 2.0, 17 September 2014

Table of Contents

1 Introduction

This document presents assurance activity evaluation results of the Microsoft Windows 10 evaluation. There are three types of assurance activities and the following is provided for each:

1. TOE Summary Specification (TSS)—an indication that the required information is in the TSS section of the Security Target
2. Guidance—a specific reference to the location in the guidance is provided for the required information
3. Test—a summary of the test procedure and result is provided for each required test activity.

This Assurance Activities Report contains sections for each functional class and family and sub-sections addressing each of the SFRs specified in the Security Target.

1.1 Evidence

[ST]	*Microsoft Windows 10 and Windows 10 Mobile Security Target*, v0.09, April 12, 2016
[Mobile Guide]	*Microsoft Windows 10 Mobile Microsoft Windows 10 Common Criteria Supplemental Admin Guidance* V1.0, February 17, 2016
[TPM 2.0 Arch]	*Trusted Platform Module Library Part 1: Architecture*, Family "2.0", Level 00, Revision 01.16, October 30, 2014
[TPM 2.0 Commands]	*Trusted Platform Module Library Part 3: Commands*, Family "2.0", Level 00, Revision 01.16, October 30, 2014

1.2 Protection Profile

[PP MDF]	*Protection Profile for Mobility Device Fundamentals*, Version 2.0, 17 September 2014

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 the [PP MDF].

2.1 Cryptographic Support (FCS)

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

FCS_CKM.1(ASYM KA) corresponds to FCS_CKM.1(1) in the [PP MDF] protection profile.

2.1.1.1 TSS Assurance Activity

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.

The TSF generates asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithms.

Table 15 Types of Keys Used by Windows in section 6.2.1 Cryptographic Algorithms and Operations identifies size and usage for keys. RSA keys can be 2048 or 3072 bits. The RSA keys are used for IPsec, TLS, Wi-Fi, and health attestations as well as signed product updates (section 6.6.6 Windows and Application Updates). Section 6.2.1 identifies the elliptical curves P-256, P-384, and P-521 for ECDSA and ECDH. The ECDSA keys are used for IPsec traffic and peer authentication. ECDH keys are used for TLS key establishment. DSA keys can be 2048 or 3072 bits. The DSA keys are used for IPsec and TLS.

See also section 2.2.6 below and [ST] section 6.3.4 VPN Client regarding TOE support of IPsec.

2.1.1.2 Guidance Assurance 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.

Section 21 Managing Cryptographic Algorithms indicates that there is no global configuration necessary for hashing algorithms, key generation schemes, or for key establishment schemes. The use of required key generation schemes and key sizes is supported and global configuration is not needed.

2.1.1.3 Test Activities (FIPS PUB 186-4 RSA Schemes)

Key Generation for FIPS PUB 186-4 RSA Schemes

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*q*
- *p and q are probably prime according to Miller-Rabin tests,*
- *GCD(p-1,e) = 1,*
- *GCD(q-1,e) = 1,*
- *2^16 <= e <= 2^256 and e is an odd integer,*
- *|p-q| > 2^(nlen/2 - 100),*
- *p >= squareroot(2)*(2^(nlen/2 -1)),*
- *q >= squareroot(2)*(2^(nlen/2 -1)),*
- *2^(nlen/2) < d < LCM(p-1,q-1),*
- *e*d = 1 mod LCM(p-1,q-1).*

Windows uses algorithm implementations validated under the Cryptographic Algorithm Validation Program (CAVP) (http://csrc.nist.gov/groups/STM/cavp/index.html). [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP RSA certificates for Windows 10: 1802, 1783, 1784, and 1798 (http://csrc.nist.gov/groups/STM/cavp/documents/dss/rsanewval.html). The relevant detail is reproduced and highlighted below.

1802	**FIPS186-4:**

	[RSASSA-PSS]: Sig(Gen): (2048 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) Sig(Ver): (2048 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) SHA Val#2886
1783	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(gen) (2048 SHA(256 , 384 , 512)) (3072 SHA(256 , 384 , 512)) SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#2373
1784	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#2871
1798	**FIPS186-4:** **186-4KEY(gen):** FIPS186-3_Fixed_e (10001) ; **PGM(ProbPrimeCondition):** 2048 , 3072 **PPTT:**(C.3) SHA Val#2886 DRBG: Val# 868

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP RSA certificates for Windows 10 Mobile: 1888, 1887, 1871, and 1889 (http://csrc.nist.gov/groups/STM/cavp/documents/dss/rsanewval.html). The relevant detail is reproduced and highlighted below.

1887	**FIPS186-4:** **[RSASSA-PSS]:** Sig(Gen): (2048 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) Sig(Ver): (1024 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(62))) (2048 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) SHA Val#3047
1888	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(gen) (2048 SHA(256 , 384 , 512)) (3072 SHA(256 , 384 , 512)) SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#3047
1871	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512))

	SHA Val#3048
1889	**FIPS186-4:** **186-4KEY(gen):** FIPS186-3_Fixed_e (10001) ; **PGM(ProbPrimeCondition):** 2048 , 3072 **PPTT:**(C.3) SHA Val#3047 DRBG: Val# 955

2.1.1.4 TSS Assurance Activity (ANSI X9.31-1998 RSA Schemes)

Key Generation for ANSI X9.31-1998 RSA Schemes

If the TSF implements the ANSI X9.311998 scheme, the evaluator shall check to ensure that the TSS describes how the keypairs are generated. In order to show that the TSF implementation complies with ANSI X9.311998, the evaluator shall ensure that the TSS contains the following information:

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

The TSF does not implement ANSI X9.31-1998 RSA schemes, and therefore this assurance activity is not applicable.

2.1.1.5 Test Activities (ECC and FCC Schemes)

Key Generation for Elliptic Curve Cryptography (ECC)

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.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP ECDSA certificates for Windows 10: 706 (http://csrc.nist.gov/groups/STM/cavp/documents/dss/ecdsanewval.html). The relevant detail is reproduced and highlighted below.

706	**FIPS186-4:** **PKG: CURVES**(P-256 P-384 P-521 ExtraRandomBits) **SigGen: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512) **SigVer: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512)) **SHS:** Val#2886 **DRBG:** Val# 868

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP RSA certificates for Windows 10 Mobile: 760. The relevant detail is reproduced and highlighted below.

760	**FIPS186-4:** **PKG: CURVES**(P-256 P-384 P-521 ExtraRandomBits) **SigGen: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512) **SigVer: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512)) **SHS:** Val#3047 **DRBG:** Val# 955

Key Generation for Finite-Field Cryptography (FFC)

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

for each FFC parameter set and key pair.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP DSA certificates for Windows 10: 983 (http://csrc.nist.gov/groups/STM/cavp/documents/dss/dsanewval.htm). The relevant detail is reproduced and highlighted below.

983	**FIPS186-4:** **PQG(gen)PARMS TESTED:** [(2048,256)SHA(256); (3072,256) SHA(256)] **PQG(ver)PARMS TESTED:** [(2048,256) SHA(256); (3072,256) SHA(256)] **Key Pair:** [(2048,256) ; (3072,256)] **SIG(gen)PARMS TESTED:** [(2048,256) SHA(256); (3072,256) SHA(256);]

	SIG(ver)PARMS TESTED: [(2048,256) SHA(256); (3072,256) SHA(256)] SHS: Val# 2886 DRBG: Val# 868

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP DSA certificates for Windows 10 Mobile: 1024. The relevant detail is reproduced and highlighted below.

1024	**FIPS186-4:** **PQG(gen)PARMS TESTED:** [(2048,256)SHA(256); (3072,256) SHA(256)] **PQG(ver)PARMS TESTED:** [(2048,256) SHA(256); (3072,256) SHA(256)] **Key Pair:** [(2048,256) ; (3072,256)] **SIG(gen)PARMS TESTED:** [(2048,256) SHA(256); (3072,256) SHA(256);] **SIG(ver)PARMS TESTED:** [(2048,256) SHA(256); (3072,256) SHA(256)] SHS: Val# 3047 DRBG: Val# 955

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

FCS_CKM.1(WLAN384) corresponds to FCS_CKM.1(2) in the [PP MDF] protection profile.

2.1.2.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes how the primitives defined and implemented by this PP are used by the TOE in establishing and maintaining secure connectivity to the wireless clients.

[ST] Section 6.2.1 Cryptographic Algorithms and Operations describes TOE use of FIPS-Approved algorithm primitives (search "Windows uses FIPS Approved algorithms"). Section 6.2.7 Networking covers the native implementation of IEEE 802.11-2012.

The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed (e.g. WPA2 certification). The evaluator shall ensure that the description of the testing methodology is of sufficient detail to determine the extent to which the details of the protocol specifics are tested.

[ST] Section 6.2.1 Cryptographic Algorithms and Operations identifies compliance for FIPS-Approved algorithms (search "FIPS validated mode is required according to the administrative guidance"). Section 6.2.7 Networking describes Wi-Fi Alliance certification (WPA2 and Wi-Fi CERTIFIED interoperability). See Wi-Fi Alliance certificates (http://www.wi-fi.org/certification):

- Microsoft Surface Pro 4: WFA62456 (http://www.wi-fi.org/content/search-page?keys=%22Surface%20Pro%204%22)
- Microsoft Lumia 950: WFA61269 (http://www.wi-fi.org/content/search-page?keys=WFA61269)
- Microsoft Lumia 950 XL: WFA62036 (http://www.wi-fi.org/content/search-page?keys=WFA62036)

- Microsoft Lumia 550: WFA62049 (http://www.wi-fi.org/content/search-page?keys=WFA62049)
- Lumia 635: WFA54130, WFA54131, and WFA56520 (http://www.wi-fi.org/content/search-page?keys=Lumia%20635)

The "model number" in WFA certificates is additional information, which could be used to identify the product. This varies from vendor to vendor based on their business. Some vendors will have different label for product and model number while others use the same for both. Microsoft uses either the name of the device or the name of the adapter on the device for the Windows mobile devices in [ST].

ST Section 6.2.7 Networking states that the TOE devices have received WPA2 certification.

The Wi-Fi Alliance web site describes the testing program (http://www.wi-fi.org see Programs). Wi-Fi alliance certification provides sufficient detail of protocol testing.

Additionally, Microsoft provides guidelines for testing hardware and software for Windows 10 and Windows 10 Mobile (https://msdn.microsoft.com/ en-us/ library/ windows/ hardware/ mt269770 (v=vs.85).aspx). Microsoft provides the Windows Hardware Lab Kit, which is documented online (https://msdn.microsoft.com/ en-us/ library/ windows/ hardware/ dn930814 (v=vs.85).aspx). The information online includes tests that verify successful completion for 802.11. The link https://msdn.microsoft.com/ en-us/ library/ windows/ hardware/ jj123746.aspx provides Microsoft tests of devices that have been certified as Wi-Fi Alliance compliant. Two requirements for this test, Device-.Network.WLAN.Base.PassWiFiAllianceCertification and Device.Network.WLAN.CSBBase.PassWiFi-AllianceCertification must be present to verify this information. This testing includes over 30 different test suites, including roaming tests, scan tests, stress tests, FIPS association tests, standby tests, Dot11W tests, and wake tests along with over variances.

For instance, the WiFi Direct Performance Tests include GoNegotiation PeerFinder, Invitation PeerFinder, and JoinExistingGo WFDPlatform tests. Each of these tests includes information on the prerequisites and details on running the test (e.g. - https://msdn.microsoft.com/ en-us/ library/ windows/ hardware/ dn293766.aspx).

The Surface Pro 4 was Wi-Fi-Certified on October 30, 2015, with the Certification ID WFA62456. This included the model number 1724. Under this certification, testing was done for:

- Security
  - WPA – Enterprise, Personal,
  - WPA2 – Enterprise, Personal,
  - EAP Type(s)
    - EAP-TLS,
    - and other protocols
- Wi-Fi CERTIFIED a
- Wi-Fi CERTIFIED b
- Wi-Fi CERTIFIED g
- Wi-Fi CERTIFIED n
- Wi-Fi CERTIFIED ac

The Lumia 950 was Wi-Fi-Certified on July 23, 2015, with the Certification ID WFA61269. This included the model number RM-1104. Under this certification, testing was done for:

- Security
  - WPA – Enterprise, Personal,

    - WPA2 – Enterprise, Personal,
    - EAP Type(s)
        - EAP-TLS,
        - and other protocols
- Wi-Fi CERTIFIED a
- Wi-Fi CERTIFIED b
- Wi-Fi CERTIFIED g
- Wi-Fi CERTIFIED n

The Lumia 950 XL was Wi-Fi-Certified on September 8, 2015, with the Certification ID WFA62036. This included the model number RM-1116. Under this certification, testing was done for:

- Security
    - WPA – Enterprise, Personal,
    - WPA2 – Enterprise, Personal,
    - EAP Type(s)
        - EAP-TLS,
        - and other protocols
- Wi-Fi CERTIFIED a
- Wi-Fi CERTIFIED b
- Wi-Fi CERTIFIED g
- Wi-Fi CERTIFIED n

The Lumia 550 was Wi-Fi-Certified on September 08, 2015, with the Certification ID WFA62049. This included the model number RM-1127. Under this certification, testing was done for:

- Security
    - WPA – Enterprise, Personal,
    - WPA2 – Enterprise, Personal,
    - EAP Type(s)
        - EAP-TLS,
        - and other protocols
- Wi-Fi CERTIFIED b
- Wi-Fi CERTIFIED g
- Wi-Fi CERTIFIED n

The Lumia 635 was Wi-Fi-Certified three times on April 8 (two models) and September 29, 2014, with the Certification IDs WFA54130, WFA54131, and WFA56520. This included the model numbers RM-974, RM-975, and RM-1078. Under this certification, testing was done for:

- Security
    - WPA – Enterprise, Personal,
    - WPA2 – Enterprise, Personal,
    - EAP Type(s)
        - EAP-TLS,
        - and other protocols
- Wi-Fi CERTIFIED b
- Wi-Fi CERTIFIED g
- Wi-Fi CERTIFIED n

2.1.2.2 Guidance Assurance Activities

None defined.

2.1.2.3 Test Activities

The evaluator shall also perform the following test using a packet sniffing tool to collect frames between a wireless access point and TOE:

Step 1: *The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.*

Step 2: *The evaluator shall configure the TOE to communicate with the access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-9 or a-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.*

Step 3: *The evaluator shall start the sniffing tool, initiate a connection between the TOE and access point, and allow the TOE to authenticate, associate and successfully complete the 4-way handshake with the access point.*

Step 4: *The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the access point and stop the sniffer.*

Step 5: *The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.*

Step 6: *The evaluator shall select the first data frame from the captured packets that was sent between the access point and TOE after the 4-way handshake successfully completed, and without the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the PTK to decrypt the data portion of the packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCII-readable text.*

Step 7: *The evaluator shall repeat Step 7 for the next 2 data frames between the TOE and access point, and without frame control value 0x4208.*

The evaluator followed the above steps to connect the TOE to the access point. The evaluator analyzed the packets without control value 0x4208 and verified that they contained ASCII-readable text.

2.1.3 Cryptographic Key Generation (WLAN) (FCS_CKM.1(3))

FCS_CKM.1(WLAN704) corresponds to FCS_CKM.1(3) in the [PP MDF] protection profile.

2.1.3.1 TSS Assurance Activity

The cryptographic primitives will be verified through assurance activities specified elsewhere in this PP. The evaluator shall verify that the TSS describes how the primitives defined and implemented by this PP are used by the TOE in establishing and maintaining secure connectivity to the wireless clients.

Section 6.2.7 Networking states that Windows (that is, Windows 10 and Windows 10 Mobile) has a native implementation of IEEE 802.11ac-2013 to provide secure wireless local area networking (Wi-Fi).

Windows uses PRF-704 to generate AES 256-bit keys, which uses the Windows RBG. Windows complies with the IEEE 802.11ac-2013 standard and interoperates with other devices that implement the standard.

The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed (e.g. WPA2 certification). The evaluator shall ensure that the description of the testing methodology is of sufficient detail to determine the extent to which the details of the protocol specifics are tested.

Section 2.1.2.1 above covers Microsoft's test methods and certifications for wireless communication. In particular, the following Wi-Fi Alliance certificates include WPA2 Enterprise and Personal:

- WFA62456 for Surface Pro 4
- WFA61269 for Lumia 950
- WFA62036 for Lumia 950 XL
- WFA62049 for Lumia 550
- WFA54130, WFA54131, and WFA56520 for Lumia 635

The underlying HMAC, SHA-384, and DRBG cryptographic algorithms are CAVP validated.

2.1.3.2 Guidance Assurance Activities

None defined.

2.1.3.3 Test Activities

The evaluator shall also perform the following test:

Step 1 *- The evaluator shall use a packet sniffing tool between the wireless access point and TOE. The evaluator shall turn on the sniffing tool and successfully connect the TOE to the access point.*

Step 2 *– The evaluator shall verify the TOE advertises 00-0F-AC:12 as a supported Authentication and Key Management (AKM) suite and either 00-0F-AC:9 or 00-0F-AC:10 as a supported cipher suite in capture 802.11 beacon and probe response messages.*

The evaluator connected the TOE to an access point and verified that is advertises 00-0F-AC:12 as a supported AKM suite and 00-0F-AC:10 as a supported cipher suite.

2.1.4 Cryptographic Key Establishment FCS_CKM.2.1(1)

FCS_CKM.2(ASYM AU) corresponds to FCS_CKM.2(1) in the [PP MDF] protection profile.

2.1.4.1 TSS Assurance Activity

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1(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.

The key establishment schemes identified in FCS_CKM.2(ASYM AU) (PP: FCS_CKM.2(1)) correspond to the key generation schemes identified in FCS_CKM.1(ASYM KA) (PP: FCS_CKM.1.1(1)): RSA, ECC, and FFC schemes. See section 2.1.1.1 above regarding key usage.

2.1.4.2 Guidance Assurance 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).

[Mobile Guide] Section 21 Managing Cryptographic Algorithms indicates that there is no global configuration necessary for hashing algorithms, key generation schemes, or for key establishment schemes. The use of required key generation schemes and key sizes is supported and global configuration is not needed.

2.1.4.3 Test Activities (SP800-56A)

Assurance 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.

Function Test

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

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

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

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

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

Validity Test

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

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

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

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP KAS certificates for Windows 10: 64 (http://csrc.nist.gov/groups/STM/cavp/documents/keymgmt/kasnewval.html). The relevant detail is reproduced and highlighted below.

64	**FFC:** (FUNCTIONS INCLUDED IN IMPLEMENTATION: DPG DPV KPG Partial Validation) **SCHEMES** [**dhEphem** (KARole(s): Initiator / Responder) (**FB:** SHA256) (**FC:** SHA256)] [**dhOneFlow** (KARole(s): Initiator / Responder) (**FB:** SHA256) (**FC:** SHA256)] [**dhStatic** (**No_KC** < KARole(s): Initiator / Responder>) (**FB:** SHA256 HMAC) (**FC:** SHA256 HMAC)] SHS Val#2886 DSA Val#983 DRBG Val#868 **ECC:** (FUNCTIONS INCLUDED IN IMPLEMENTATION: DPG DPV KPG Partial Validation Key Regeneration) **SCHEMES** [**EphemeralUnified** (**No_KC** < KARole(s): Initiator / Responder>) (**EC:** P-256 SHA256 HMAC) (**ED:** P-384 SHA384 HMAC) (**EE:** P-521

	HMAC (SHA512, HMAC_SHA512)))] [**OnePassDH** (**No_KC** < KCRole(s): Initiator Responder>) (**EB:**) (**EC:** P-256 SHA256 HMAC) (**ED:** P-521 SHA384 HMAC) (**EE:** P-521 HMAC (SHA512, HMAC_SHA512))] [**StaticUnified** (**No_KC** < KARole(s): Initiator / Responder>) (**EC:** P-256 SHA256 HMAC) (**ED:** P-384 SHA384 HMAC) (**EE:** P-521 HMAC (SHA512, HMAC_SHA512))] SHS Val#2886 ECDSA Val#706 DRBG Val#868

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP KAS certificates for Windows 10 Mobile: 72. The relevant detail is reproduced and highlighted below.

72	**FFC:** (FUNCTIONS INCLUDED IN IMPLEMENTATION: DPG DPV KPG Partial Validation) **SCHEMES** [**dhEphem** (KARole(s): Initiator / Responder) (**FB:** SHA256) (**FC:** SHA256)] [**dhOneFlow** (KARole(s): Initiator / Responder) (**FB:** SHA256) (**FC:** SHA256)] [**dhStatic** (**No_KC** < KARole(s): Initiator / Responder>) (**FB:** SHA256 HMAC) (**FC:** SHA256 HMAC)] SHS Val#3047 DSA Val#1024 DRBG Val#955 **ECC:** (FUNCTIONS INCLUDED IN IMPLEMENTATION: DPG DPV KPG Partial Validation Key Regeneration) **SCHEMES** [**EphemeralUnified** (**No_KC** < KARole(s): Initiator / Responder>) (**EC:** P-256 SHA256 HMAC) (**ED:** P-384 SHA384 HMAC) (**EE:** P-521 HMAC (SHA512, HMAC_SHA512)))] [**OnePassDH** (**No_KC** < KARole(s): Initiator / Responder>) (**EB:**) (**EC:** P-256 SHA256 HMAC) (**ED:** P-521 SHA384 HMAC) (**EE:** P-521 HMAC (SHA512, HMAC_SHA512))] [**StaticUnified** (**No_KC** < KARole(s): Initiator / Responder>) (**EC:** P-256 SHA256 HMAC) (**ED:** P-384 SHA384 HMAC) (**EE:** P-521 HMAC (SHA512, HMAC_SHA512))] SHS Val#3047 ECDSA Val#760 DRBG Val#955

2.1.4.4 TSS Assurance Activity

SP800-56B Key Establishment Schemes

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.

[ST] section 6.2.1 Cryptographic Algorithms and Operations states when Windows needs to establish an RSA-based shared secret key it can act both as a sender or recipient.

SP800-56B Key Establishment Schemes

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.

[ST] section 6.2.1 Cryptographic Algorithms and Operations states any decryption errors which occur during key establishment are presented to the user at a highly abstracted level, such as a failure to connect.

2.1.4.5 Guidance Assurance Activities (SP800-56B)

SP800-56B Key Establishment Schemes

None defined.

2.1.4.6 Test Activities (SP800-56B)

SP800-56B Key Establishment Schemes

If the TOE acts as a sender, the following assurance 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 KTS-OAEP 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.*

If the TOE acts as a receiver, the following assurance 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 KTS-OAEP 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.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CVL certificates for Windows 10: 576 (http://csrc.nist.gov/groups/STM/cavp/documents/components/componentnewval.html). The relevant detail is reproduced and highlighted below.

576	**RSADP**: (Mod2048)

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP CVL certificates for Windows 10 Mobile: 663. The relevant detail is reproduced and highlighted below.

663	**RSADP:** (Mod2048)

SP800-56B Key Establishment Schemes

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.

[ST] does not claim Key Transport Scheme with Optimal Asymmetric Encryption Padding support.

2.1.5 Cryptographic Key Distribution (WLAN) FCS_CKM.2.1(2)

FCS_CKM.2(GTK) corresponds to FCS_CKM.2(2) in the [PP MDF] protection profile.

2.1.5.1 TSS Assurance Activity

The evaluator shall check the TSS to ensure that it describes how the GTK is unwrapped prior to being installed for use on the TOE using the AES implementation specified in this PP.

[ST] Section 6.2.7 Networking describes AES Key Wrap in accordance with NIST SP 800-38F using KW mode. (Search "Windows implements key wrapping and unwrapping".) In addition, Section 6.2.8 SFR Mapping provides a link to Windows' native implementation of IEEE 802.11 (http://msdn.microsoft.com/en-us/library/windows/hardware/ff556022(v=vs.85).aspx).

2.1.5.2 Guidance Assurance Activities

None defined.

2.1.5.3 Test Activities

The evaluator shall also perform the following test using a packet sniffing tool to collect frames between a wireless access point and TOE (which may be performed in conjunction with the assurance activity for FCS_CKM.1.1(2):

Step 1: *The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.*

Step 2: *The evaluator shall configure the TOE to communicate with the access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-9 or a-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.*

Step 3: *The evaluator shall start the sniffing tool, initiate a connection between the TOE and access point, and allow the TOE to authenticate, associate and successfully complete the 4-way handshake with the access point.*

Step 4*: The evaluator shall set a timer for 1 minute, at the end of which the evaluator shall disconnect the TOE from the access point and stop the sniffer.*

Step 5: *The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and derive the PTK and GTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.*

Step 6: *The evaluator shall select the first data frame from the captured packets that was sent between the access point and TOE after the 4-way handshake successfully completed, and with the frame control value 0x4208 (the first 2 bytes are 08 42). The evaluator shall use the GTK to decrypt the data portion of the selected packet as specified in IEEE 802.11-2012, and shall verify that the decrypted data contains ASCII-readable text.*

Step 7: *The evaluator shall repeat Step 7 for the next 2 data frames with frame control value 0x4208.*

The evaluator followed the above steps to connect the TOE to the access point. The evaluator analyzed the packets with control value 0x4208 and verified that they contained ASCII-readable text.

2.1.6 Cryptographic Key Support (REK) FCS_CKM_EXT.1

2.1.6.1 TSS Assurance Activity

The evaluator shall review the TSS to determine that a REK is supported by the product, that the TSS includes a description of the protection provided by the product for a REK, and that the TSS includes a description of the method of generation of a REK.

[ST] section 6.2.4 Encrypting the Device with BitLocker identifies the Storage Primary Seed (SPS) in the Trusted Platform Module (TPM) as the REK. The TPM generates the SPS using the TPM RBG as described in section 6.2.4.

The evaluator shall verify that the description of the protection of a REK describes how any reading, import, and export of that REK is prevented. (For example, if the hardware protecting the REK is

removable, the description should include how other devices are prevented from reading the REK.)

[ST] Section 6.2.3 Trusted Platform Module states "The TPM also provides protections that prevent the export of TPM keys and cryptographic data, such as the SPS and SRK." Specifically, section 6.2.3 Trusted Platform Module states: "Like other cryptographic data within the TPM, the private portion of a key created in a TPM is never exposed to any other component, software, process, or user." In addition, section 6.2.4 Encrypting the Device with BitLocker states: "… the REK is isolated from operating system and applications, thus preventing reading and exporting the plaintext representation of the REK." [TPM 2.0 Arch] specifies a TPM uses the SPS to derive Storage Root Keys (SRK) (section 14.3.4) and the TPM never stores a Primary Seed off the TPM in any form (section 14.1).

The evaluator shall verify that the TSS describes how encryption/decryption/derivation actions are isolated so as to prevent applications and system-level processes from reading the REK while allowing encryption/decryption/derivation by the key.

[TPM 2.0 Arch] specifies TPM processor, volatile memory, and persistent storage are isolated from platform hardware and software (section 9.3). Consequently, Windows can only access TPM services through the well-defined APIs provided by the TPM. [TPM 2.0 Arch] states the TPM never stores a Primary Seed off the TPM in any form.

If "hardware-isolated" is selected and REK(s) are isolated from the rich OS by a separate processor execution environment, the evaluator shall verify that the description includes how the rich OS is prevented from accessing the memory containing REK key material, which software is allowed access to the REK, how any other software in the execution environment is prevented from reading that key material, and what other mechanisms prevent the REK key material from being written to shared memory locations between the rich OS and the separate execution environment.

The TOE in the evaluated configuration conform to TPM standard 2.0 as identified in [ST] section 1.1 Security Target, TOE, and Common Criteria (CC) Identification. [TPM 2.0 Arch] specifies TPM processor, volatile memory, and persistent storage are isolated from platform hardware and software (section 9.3). Consequently, Windows can only access TPM services through the well-defined APIs provided by the TPM. [TPM 2.0 Arch] states the TPM never stores a Primary Seed off the TPM in any form.

If key derivation is performed using a REK, the evaluator shall ensure that the TSS description includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108. (Additional key expansion algorithms are defined in other NIST Special Publications.)

[ST] Section 6.2.4 Encrypting the Device with BitLocker states the TPM generates the SRK during initialization. [TPM 2.0 Arch] section 11.4.8 describes SRK generation from the SPS, since the SRK is a Primary Key.

The evaluator shall verify that the generation of a REK meets the FCS_RBG_EXT.1.1 and FCS_RBG_EXT.1.2 requirements:

- If REK(s) is/are generated on-device, the TSS shall include a description of the generation mechanism including what triggers a generation, how the functionality described by FCS_RBG_EXT.1 is invoked, and whether a separate instance of the RBG is used for REK(s).
- If REK(s) is/are generated off-device, the TSS shall include evidence that the RBG meets FCS_RBG_EXT.1.2. This will likely a second set of RBG documentation equivalent to the documentation provided for the RBG assurance activities. In addition, the TSS shall describe the manufacturing process that prevents the device manufacturer from accessing any REKs.

[ST] Section 6.2.4 Encrypting the Device with BitLocker states SPS is created as part of the BitLocker initialization process. Windows causes the TPM to generate the SPS.

The second bullet does not apply, since Windows causes the TPM to generate the SPS.

2.1.6.2 Guidance Assurance Activities

None defined.

2.1.6.3 Test Activities

None defined.

2.1.7 Cryptographic Key Random Generation (FCS_CKM_EXT.2)

FCS_CKM_EXT.2(128) and FCS_CKM_EXT.2(256) corresponds to FCS_CKM_EXT.2 in the [PP MDF] protection profile.

2.1.7.1 TSS Assurance Activity

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 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 data.

[ST] section 6.2.6 Protecting Data with DPAPI states the Windows RBG (as described by FCS_RBG_EXT.1) generates a DPAPI Master Secret which is used as input into an AES function along with an initialization vector and encryption key, both of which are based on the user's password, to generate the encrypted DPAPI Master Secret. The DPAPI Master Secret is a kind of DEK and the password-based encryption key, which protects the DPAPI Master Secret, is a kind of KEK.

[ST] sections 6.2.4 Encrypting the Device with BitLocker, 6.2.5 Key Storage, and 6.2.6 Protecting Data with DPAPI describe how Windows uses key and data encryption. The following list summarizes the key hierarchy and identifies the type (KEK or DEK) of each key.

1. TPM Storage Primary Seed is the REK, which derives the Storage Root
2. TPM Storage Root Key is a KEK, which encrypts intermediate keys.
3. Intermediate keys are KEKs, which encrypt Volume Master Key (VMK)
4. Volume Master Key is KEK, which encrypt partition Full Volume Encryption Key (FVEK), which is DEK.
5. DPAPI provides encryption of software-based key storage in addition to BitLocker
    a. DPAPI password-based encryption key is a KEK, which encrypts the DPAPI Master Secret
    b. The DPAPI Master Secret is a KEK, which encrypts data encryption keys
    c. Data encryption keys are DEKs, which encrypt key storage (one data encryption key per user)

FCS_CKM_EXT.2(128) and FCS_CKM_EXT.2(256) apply to the following DEKs: FVEK, DPAPI and data encryption key (one per user).

[ST] Section 6.2.4 Encrypting the Device with BitLocker indicates that the administrator configures BitLocker to use either a 128-bit or 256-bit FVEK for Windows 10. The FVEK for Windows 10 Mobile is always 128 bits.

[Mobile Guide] describes managing Windows 10 volume encryption in section 7 Managing Volume Encryption. The section and documentation for manage-bde covers parameter "encryptionmethod," which sets AES key size. (See http://technet.microsoft.com/en-us/library/ff829849(v=ws.10).aspx.) By default AES128 encryption is used by the manage-bde command when enabling BitLocker for Windows 10 – the AES256 algorithm should be used instead. In addition, [Mobile Guide] describes how to configure the TPM, PIN authorization factor, and Enhanced PIN capabilities that must be used in the evaluated configuration.

DPAPI data encryption keys are 256-bit keys.

2.1.7.2 Guidance Assurance Activities

None defined.

2.1.7.3 Test Activities

None defined.

2.1.8 Cryptographic Key Encryption Keys (FCS_CKM_EXT.3)

2.1.8.1 TSS Assurance Activity

The evaluator shall examine the key hierarchy TSS to ensure that the formation of all KEKs is described and that the key sizes match that described by the ST author.

The evaluator shall review the TSS to verify that it contains a description of the PBKDF use to derive KEKs. This description must include the size and storage location of salts. This activity may be performed in combination with that for FCS_COP.1(5).

See section 2.1.7.1 above in Cryptographic Key Random Generation (FCS_CKM_EXT.2) for a summary of the key hierarchy. Table 1 below summarizes the sizes of KEKs. See 2.1.18.1 below in Initialization Vector Generation (FCS_IV_EXT.1) for cipher modes.

[ST] section 6.2.6 Protecting Data with DPAPI describes the PBKDF2 function that takes a result of a one-way function computation of the user's password to generate the password encryption key that protects the DPAPI Master Secret (a kind of KEK). Section 6.2.1 Cryptographic Algorithms and Operations includes PBKDF details. (Search "The HMAC function forms the basis".) Section 6.2.6 states Windows will also combine the DPAPI Master Secret along with a salt value which will be used as an encryption key to protect user data, such as a private key. The salt value is stored with the Master Secret.

If the KEK is generated, the evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT.1 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 data.

Based on the check made for FCS_CKM_EXT.2 above, FCS_CKM_EXT.3 applies to the following KEKs: Storage Primary Seed (also REK), Storage Root Key, intermediate keys, VMK, password-based encryption key, DPAPI Master Secret, Device User Credential Keys, and User's public/private key pairs. [ST] sections 6.2.4 Encrypting the Device with BitLocker, 6.2.6 Protecting Data with DPAPI, and 6.4.1 Protecting User Data identify the size and formation method of each KEK. Table 1 summarizes the size and formation method information.

Table 1 KEK Formation Method and Key Size

KEK	Formation Method	Size
Storage Primary Seed	TPM RBG	256
Storage Root Key	TPM RSA	2048
Intermediate Keys	XOR (Enhanced PIN) and RBG	256
VMK	RBG	256
DPAPI password-based encryption key	PBKDF2 from passphrase	256
DPAPI Master Secret	RBG	At least 256
Device User Credential Keys	RBG	256
User's public/private key pairs	RSA key generation	2048

NIAP TD0038 accommodates use of a REK with key strength of 112 bits (for example, a 2048-bit RSA key). In this case, the strengths of keys in a chain are not non-increasing. The security strength of a chain is limited by the key with the least strength and not necessarily the last key in the chain. Thus, security strength of data encryption is limited by the strengths of the Storage Root Key, FVEK, Enhanced PIN, and password, since all other keys are at least 256 bits.

If the KEK is formed from a combination, the evaluator shall verify that the TSS describes the method of combination and that this method is either an XOR, a KDF, or encryption. If a KDF is used, the evaluator shall ensure that the TSS description includes a description of the key derivation function and shall verify the key derivation uses an approved derivation mode and key expansion algorithm according to SP 800-108. (Additional key expansion algorithms are defined in other NIST Special Publications.)

[ST] section 6.2.4 Encrypting the Device with BitLocker states that the FVEK and intermediate keys are all generated by the Windows RBG or by combining intermediate keys as described in FCS_CKM_EXT.3.

2.1.8.2 Guidance Assurance Activities

None defined.

2.1.8.3 Test Activities

None defined.

2.1.9 Cryptographic Key Destruction (FCS_CKM_EXT.4)

2.1.9.1 TSS Assurance Activity

The evaluator shall check to ensure the TSS lists each type of plaintext key material (DEKs, software-based key storage, KEKs, trusted channel keys, passwords, etc.) and its origin and storage location.

[ST] section 6.2.5 Key Storage discusses the key material. "The encrypted FVEK, VMK, and Intermediate Key are stored on disk as metadata on the storage volume, however the metadata is stored outside of the mounted NTFS volume and so these are never transmitted outside the device, which the boundary of the cryptographic module in this evaluation."

Section 6.2.1 Cryptographic Algorithms and Operations lists public, private, and secret keys used for TLS, and Wi-Fi in Table 15 Types of Keys Used by Windows. Windows stores persistent public, private, and secret keys in the NTFS file system as described in section 6.2.5. Storage protection includes DPAPI protection as referenced in section 6.2.5 and described in section 6.2.6 Protecting Data with DPAPI.

See also section 2.2.6 below and [ST] section 6.3.4 VPN Client regarding TOE support of IPsec.

Section 6.2.4 Encrypting the Device with BitLocker states that the unencrypted VMK is zeroized after it is (1) used to encrypt the FVEK and (2) encrypted by an intermediate key.

The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write"). For block erases, the evaluator shall also ensure that the block erase command used is listed and shall verify that the command used also addresses any copies of the plaintext key material and that may be created in order to optimize the use of flash memory.

[ST] section 6.2.4 Encrypting the Device with BitLocker states: "The unencrypted VMKs are zeroized after they are (1) used to encrypt the FVEK and (2) encrypted by an intermediate key. The other keys are also zeroized from volatile memory in the process of generating the VMK. When Windows shuts down normally or goes into hibernation, Windows will zeroize the FVEK as part of shutdown. In the event of a system crash, the BitLocker Crash Dump Filter will zeroize the FVEK in order to prevent the FVEK from being included in the crash dump file."

Section 6.2.5 Key Storage states: "Destruction of keys/secrets imported into the secure key storage by applications is conducted automatically by the modern application environment after the keys/secrets are no longer in use."

"Private keys are protected on disk using DPAPI and BitLocker encryption and access is restricted using the Windows Discretionary Access Control Policy. When a Windows Store Application is deleted the local private keys imported by that app are deleted. All private keys are destroyed when a wipe operation is performed on a device. Local administrators can also perform a wipe on their Windows device to destroy all the keys or secrets. The IT administrator can perform a wipe operation of the enrolled device to destroy the keys."

Section 6.2.1 Cryptographic Algorithms and Operations addresses clearing of public, private, and secret keys used for TLS and Wi-Fi in Table 15 Types of Keys Used by Windows. The keys are cleared as specified in section 6.2.1. The description covers both persistent keys (that is, keys in non-volatile memory) and ephemeral keys (that is, keys in volatile memory). Section 6.2.1 describes deleting persistent keys in non-volatile flash and overwriting ephemeral keys in volatile memory. Windows does not store plaintext keys in flash.

See also section 2.2.6 below and [ST] section 6.3.4 VPN Client regarding TOE support of IPsec.

2.1.9.2 Guidance Assurance Activities

None defined.

2.1.9.3 Test Activities

Assurance 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.*

For each software and firmware key clearing situation (including on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, when transitioning to the locked state, and possibly including immediately after use, while in the locked state) the evaluator shall repeat the following tests. Note that at this time hardware-bound keys are explicitly excluded from testing.

***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 attached the TOE to a debugger and used a tool developed by the evaluation team to create and clear an encryption key. The evaluator dumped memory before and after the key was cleared and verified that no traces of the key remained and the memory was zeroized.

***Test 2:** In cases where the TOE is implemented in firmware and operates in a limited operating environment that does not allow the use of debuggers, the evaluator shall utilize a simulator for the TOE on a general purpose operating system. The evaluator shall provide a rationale explaining the instrumentation of the simulated test environment and justifying the obtained test results.*

N/A—The TOE is not implemented in firmware.

2.1.10 TSF Wipe (FCS_CKM_EXT.5)

2.1.10.1 TSS Assurance Activity

The evaluator shall check to ensure the TSS describes how the device is wiped; and the type of clearing procedure that is performed (cryptographic erase or overwrite) and, if overwrite is

performed, the overwrite procedure (overwrite with zeros, overwrite three or more times by a different alternating pattern, overwrite with random pattern, or block erase).

[ST] section 6.3.2 Data at Rest Protection describes wiping the device by deleting the authorization factors that unlock the device. (Search "decides to wipe the device"). The section covers how Windows deletes the authorization factors. The clearing process is performed by first overwriting the metadata with zeros followed by a read-verify. [ST] section 6.2.5 Key Storage adds that wiping destroys all private keys and secrets.

If different types of memory are used to store the data 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, "data stored on flash are cleared by overwriting once with zeros, while data stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

Windows stores persistent keys encrypted in flash memory as described in [ST] section 6.2.5 Key Storage. (Search "The encrypted FVEK, VMK, and Intermediate Key are stored on disk".) [ST] section 6.3.2 Data at Rest Protection indicates the BitLocker metadata (which includes the authorization factors that unlock the device) is deleted from flash. The wiping of the BitLocker metadata from flash memory is performed by first overwriting the metadata with zeros, followed by a read-verify.

[ST] section 6.2.1 Cryptographic Algorithms and Operations indicates that the TOE deletes persistent keys by deleting the storage block. (Search "when it is necessary to delete a persistent key from flash memory".)

2.1.10.2 Guidance Assurance Activities

None defined.

2.1.10.3 Test Activities

Assurance Activity Note: *The following test may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

The evaluator shall perform one of the following tests. The test before and after the wipe command shall be identical. This test shall be repeated for each type of memory used to store the data to be protected.

Method 1 for File-based Methods:

Test: The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a user data (protected data or sensitive data) file, for example, by using an application. The evaluator shall use a tool provided by the developer to examine this data stored in memory (for example, by examining a decrypted files). The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF_EXT.1. The evaluator shall use a tool provided by the developer to examine the same data location in memory to verify that the data has been wiped

according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

Method 2 for Volume-based Methods:

Test: The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create a unique data string, for example, by using an application. The evaluator shall use a tool provided by the developer to search decrypted data for the unique string. The evaluator shall initiate the wipe command according to the AGD guidance provided for FMT_SMF_EXT.1. The evaluator shall use a tool provided by the developer to search for the same unique string in decrypted memory to verify that the data has been wiped according to the method described in the TSS (for example, the files are still encrypted and cannot be accessed).

The evaluator encrypted the TOE and created and saved a file to long term storage. The evaluator viewed the unencrypted content and initiated a wipe command. After the wipe, the evaluator viewed the drive contents at the same offset that the file was stored and verified that the data was wiped and no reference to it remained.

2.1.11 Cryptographic Salt Generation (FCS_CKM_EXT.6)

2.1.11.1 TSS Assurance Activity

The evaluator shall verify that the TSS contains a description regarding the salt generation, including which algorithms on the TOE require salts. The evaluator shall confirm that the salt is generating using an RBG described in FCS_RBG_EXT.1. For PBKDF derivation of KEKs, this assurance activity may be performed in conjunction with FCS_CKM_EXT.3.2.

[ST] section 6.2.1 Cryptographic Algorithms and Operations states “When Windows requires the use of a salt it uses the Windows RBG.” Windows uses salt to generate DPAPI keys protecting user data (section 6.2.6 Protecting Data with DPAPI). [ST] section 6.2.8 SFR Mapping adds “When Windows needs to generate a salt for any kind of signature generation or key agreement, and to derive a key from a passphrase, it uses the Windows random bit generator.”

2.1.11.2 Guidance Assurance Activities

None defined.

2.1.11.3 Test Activities

None defined.

2.1.12 Cryptographic Operation (FCS_COP.1(1))

FCS_COP.1(SYM) corresponds to FCS_COP.1(1) in the [PP MDF] protection profile.

2.1.12.1 TSS Assurance Activity

None defined.

2.1.12.2 Guidance Assurance Activities

None defined.

2.1.12.3 Test Activities

Assurance 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.*

AES-CBC Tests

AES-CBC Known Answer Tests

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

KAT-1. *To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros.*

Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

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

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

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

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of

key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

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

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

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);

3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

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.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0

	- 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

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.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES

	Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

AES-CCM Tests

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

128 bit and 256 bit keys

Two payload lengths. *One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).*

Two or three associated data lengths*. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 216 bytes, an associated data length of 216 bytes shall be tested.*

Nonce lengths. *All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.*

Tag lengths. *All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.*

To test the generation-encryption functionality of AES-CCM, the evaluator shall perform the following four tests:

Test 1. *For EACH supported key and associated data length and ANY supported payload, nonce 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.*

Test 2. *For EACH supported key and payload length and ANY supported associated data, nonce 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.*

Test 3. *For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator shall supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.*

Test 4. *For EACH supported key and tag length and ANY supported associated data, payload 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.*

To determine correctness in each of the above tests, the evaluator shall compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

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 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator shall supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator shall use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AES-CCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16)

	CMAC (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

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

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

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

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96)

	(KS: AES_256(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

XTS-AES Test

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

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);

	CFB128 (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator shall test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

128 and 256 bit key encryption keys (KEKs)

***Three plaintext lengths**. One of the plaintext lengths shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).*

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator shall use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator shall test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

The evaluator shall test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).

One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).

The evaluator shall test the authenticated-decryption functionality of AES-KWP using the same test as

for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10: 3476, 3497, 3498, and 3507 (http://csrc.nist.gov/groups/STM/cavp/documents/aes/aesval.html). The relevant detail is reproduced and highlighted below.

3476	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3497	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verfication) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_256**(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **IV Lengths Tested:** (0 , 0) ; **96BitIV_Supported GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3498	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3497
3507	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3497

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP AES certificates for Windows 10 Mobile: 3630, 3629, 3653, and 3652. The relevant detail is reproduced and highlighted below.

3630	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256);
3629	**ECB** (e/d; 128 , 192 , 256); **CBC** (e/d; 128 , 192 , 256); **CFB8** (e/d; 128 , 192 , 256); **CFB128** (e/d; 128 , 192 , 256); **CTR** (int only; 128 , 192 , 256) **CCM (KS: 128 , 192 , 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 7 8 9 10 11 12 13 **(Tag Length(s)**: 4 6 8 10 12 14 16) **CMAC** (Generation/Verification) **(KS: 128;** Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 192**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **(KS: 256**; Block Size(s): Full / Partial ; **Msg Len(s)** Min: 0 Max: 2^16 ; **Tag Len(s)** Min: 0 Max: 16) **GCM (KS: AES_128**(e/d) Tag Length(s): 128 120 112 104 96) **(KS: AES_192**(e/d) Tag Length(s): 128 120 112 104 96)

	(KS: AES_256(e/d) Tag Length(s): 128 120 112 104 96) **IV Generated:** (Externally) ; **PT Lengths Tested:** (0 , 1024 , 8 , 1016) ; **AAD Lengths tested:** (0 , 1024 , 8 , 1016) ; **96BitIV_Supported** **GMAC_Supported** **XTS((KS: XTS_128**((e/d) (f)) **KS: XTS_256**((e/d) (f))
3653	**CCM (KS: 256**) **(Assoc. Data Len Range**: 0 - 0 , 2^16) **(Payload Length Range**: 0 - 32 (**Nonce Length(s)**: 12 **(Tag Length(s)**: 16) AES Val#3629
3652	**KW** (AE , AD , AES-128 , AES-192 , AES-256 , FWD , 128 , 256 , 192 , 320 , 2048) AES Val#3629

2.1.13 Hashing Algorithms (FCS_COP.1(2))

FCS_COP.1(HASH) corresponds to FCS_COP.1(2) in the [PP MDF] protection profile.

2.1.13.1 TSS Assurance Activity

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

[ST] associates hash functions with IKE and TLS (Section 6.2.1 Cryptographic Algorithms and Operations (search "as well as hashing functions")) as well as cryptographic services HMAC, RSA, DSA, ECDSA, Diffie-Hellman, elliptic curve Diffie-Hellman, and Dual EC DRBG (Section 6.2.1 Cryptographic Algorithms and Operations (search "Hashing is used")).

2.1.13.2 Guidance Assurance 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 hash sizes is present.

[Mobile Guide] Section 21 Managing Cryptographic Algorithms indicates that there is no global configuration necessary for hashing algorithms, key generation schemes, or for key establishment schemes. The use of required hashing algorithms, key generation schemes and key sizes is supported and global configuration is not needed.

2.1.13.3 Test Activities

The TSF hashing functions can be implemented in one of two modes. The first mode is the byteoriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bitoriented 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 bitoriented vs. the byteoriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF

and used to satisfy the requirements of this PP.

Assurance 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.*

Short Messages Test Bitoriented Mode

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

Short Messages Test Bit-oriented Mode

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

Short Messages Test Byte-oriented Mode

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

Selected Long Messages Test Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where $1 \leq i \leq m$. 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.

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.

Pseudorandomly Generated Messages Test

This test is for byteoriented 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.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP SHA certificates for

Windows 10: 2886 and 2871 (http://csrc.nist.gov/groups/STM/cavp/documents/shs/shaval.htm). The relevant detail is reproduced and highlighted below.

2886	**SHA-1** (BYTE-only) **SHA-256** (BYTE-only) **SHA-384** (BYTE-only) **SHA-512** (BYTE-only)
2871	**SHA-1** (BYTE-only) **SHA-256** (BYTE-only) **SHA-384** (BYTE-only) **SHA-512** (BYTE-only)

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP SHA certificates for Windows 10 Mobile: 3048 and 3047. The relevant detail is reproduced and highlighted below.

3048	**SHA-1** (BYTE-only) **SHA-256** (BYTE-only) **SHA-384** (BYTE-only) **SHA-512** (BYTE-only)
3047	**SHA-1** (BYTE-only) **SHA-256** (BYTE-only) **SHA-384** (BYTE-only) **SHA-512** (BYTE-only)

2.1.14 Signature Algorithms (FCS_COP.1(3))

FCS_COP.1(SIGN) corresponds to FCS_COP.1(3) in the [PP MDF] protection profile.

2.1.14.1 TSS Assurance Activity

None defined.

2.1.14.2 Guidance Assurance Activities

None defined.

2.1.14.3 Test Activities

Assurance 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.

ECDSA Algorithm Tests

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.

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

Signature Generation Test

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

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

Signature Verification Test

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

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP ECDSA certificates for Windows 10: 706 (http://csrc.nist.gov/groups/STM/cavp/documents/dss/ecdsanewval.html). The relevant detail is reproduced and highlighted below.

706	**FIPS186-4:** **PKG: CURVES**(P-256 P-384 P-521 ExtraRandomBits) **SigGen: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512) **SigVer: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512)) **SHS:** Val#2886 **DRBG:** Val# 868

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP RSA certificates for Windows 10 Mobile: 760. The relevant detail is reproduced and highlighted below.

760	**FIPS186-4:** **PKG: CURVES**(P-256 P-384 P-521 ExtraRandomBits) **SigGen: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512) **SigVer: CURVES**(P-256: (SHA-256) P-384: (SHA-384) P-521: (SHA-512)) **SHS:** Val#3047 **DRBG:** Val# 955

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP RSA certificates for Windows 10: 1802, 1783, 1784, and 1798 (http://csrc.nist.gov/groups/STM/cavp/documents/dss/rsanewval.html). The relevant detail is reproduced and highlighted below.

1802	**FIPS186-4:** **[RSASSA-PSS]:** Sig(Gen): (2048 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) Sig(Ver): (2048 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) SHA Val#2886
1783	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(gen) (2048 SHA(256 , 384 , 512)) (3072 SHA(256 , 384 , 512)) SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#2373
1784	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#2871
1798	**FIPS186-4:** **186-4KEY(gen):** FIPS186-3_Fixed_e (10001) ; **PGM(ProbPrimeCondition):** 2048 , 3072 **PPTT:**(C.3) SHA Val#2886 DRBG: Val# 868

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP RSA certificates for Windows 10 Mobile: 1888, 1887, 1871, and 1889. The relevant detail is reproduced and highlighted below.

1887	**FIPS186-4:** **[RSASSA-PSS]:** Sig(Gen): (2048 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(224 , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64)))

	Sig(Ver): (1024 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(62))) (2048 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) (3072 SHA(1 SaltLen(20) , 256 SaltLen(32) , 384 SaltLen(48) , 512 SaltLen(64))) SHA Val#3047
1888	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(gen) (2048 SHA(256 , 384 , 512)) (3072 SHA(256 , 384 , 512)) SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#3047
1871	**FIPS186-4:** **ALG[RSASSA-PKCS1_V1_5]** SIG(Ver) (1024 SHA(1 , 256 , 384 , 512)) (2048 SHA(1 , 256 , 384 , 512)) (3072 SHA(1 , 256 , 384 , 512)) SHA Val#3048
1889	**FIPS186-4:** **186-4KEY(gen):** FIPS186-3_Fixed_e (10001) ; **PGM(ProbPrimeCondition):** 2048 , 3072 **PPTT:**(C.3) SHA Val#3047 DRBG: Val# 955

2.1.15 Keyed Hash Algorithms (FCS_COP.1(4))

FCS_COP.1(HMAC) corresponds to FCS_COP.1(4) in the [PP MDF] protection profile.

2.1.15.1 TSS Assurance Activity

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

[ST] Table 13 HMAC Characteristics in Section 6.2.1 Cryptographic Algorithms and Operations identifies the HMAC key length, hash function used, block size, and output MAC length used for each algorithm.

2.1.15.2 Guidance Assurance Activities

None defined.

2.1.15.3 Test Activities

Assurance 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.*

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.

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP HMAC certificates for Windows 10: 2233 (http://csrc.nist.gov/groups/STM/cavp/documents/mac/hmacval.html). The relevant detail is reproduced and highlighted below.

2233	**HMAC-SHA1 (Key Sizes Ranges Tested:** KS<BS KS=BS KS>BS) SHS Val#2886
	HMAC-SHA256 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#2886
	HMAC-SHA384 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#2886
	HMAC-SHA512 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHSVal#2886

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP HMAC certificates for Windows 10 Mobile: 2381. The relevant detail is reproduced and highlighted below.

2381	**HMAC-SHA1 (Key Sizes Ranges Tested:** KS<BS KS=BS KS>BS) SHS Val#3047
	HMAC-SHA256 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#3047
	HMAC-SHA384 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHS Val#3047
	HMAC-SHA512 (Key Size Ranges Tested: KS<BS KS=BS KS>BS) SHSVal#3047

2.1.16 Password-Based Key Derivation Functions (FCS_COP.1(5))

FCS_COP.1(PBKD64) and FCS_COP.1(PBKDARM) correspond to FCS_COP.1(5) in the [PP MDF] protection profile.

2.1.16.1 TSS Assurance Activity

The evaluator shall check that the TSS describes the method by which the password is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself.

[ST] section 6.2.1 Cryptographic Algorithms and Operations describes how Windows inputs a password to the PBKDF (search "text string without any optional padding or blocking"). The HMAC functions specified in FCS_COP.1.1(PBKD64) and FCS_COP.1(PBKDARM) are the same as the HMAC functions specified in FCS_COP.1(HMAC) and consistent with the hash functions specified in FCS_COP.1(HASH). Section 6.2.1 identifies SHA-512 as the function used by DPAPI.

The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function and is the same length as the KEK as specified in FCS_CKM_EXT.3.

[ST] section 6.2.1 Cryptographic Algorithms and Operations states that Windows implements SP 800-132, with SHA-512 included in the list of hash functions. (Search 'the Windows implementation of SP 800-132".) Section 6.2.6 Protecting Data with DPAPI describes generating the encryption key from a user's password with PBKDF2 based on a one-way function (that is, HMAC identified in section 6.2.1).

For the NIST SP 800-132-based conditioning of the passphrase, the required assurance activities will be performed when doing the assurance activities for the appropriate requirements (FCS_COP.1.1(4)). If any manipulation of the key is performed in forming the submask that will be used to form the KEK, that process shall be described in the TSS.

Section 6.2.6 Protecting Data with DPAPI describes how Windows uses the output of PBKDF2 to create an initialization vector and encryption key for encrypting the DPAPI master secret

The evaluator shall verify that the iteration count for PBKDFs performed by the TOE comply with NIST SP 800-132 by ensuring that the TSS contains a description of the estimated time required to derive key material from passwords and how the TOE increases the computation time for password-based key derivation (including but not limited to increasing the iteration count).

[ST] section 6.2.6 presents rationale that for Windows, the time needed to derive key material form passwords is irrelevant to both online and offline attacks. Windows exceeds the iteration count recommended in SP 800-132 (1000 iterations) for both ARM implementations (3300 iterations) and 64-bit editions (8000 iterations).

2.1.16.2 Guidance Assurance Activities

None defined.

2.1.16.3 Test Activities

No explicit testing of the formation of the submask from the input password is required.

2.1.17 Extended: HTTPS Protocol (FCS_HTTPS_EXT.1)

2.1.17.1 TSS Assurance Activity

None Defined.

2.1.17.2 Guidance Assurance Activities

None Defined.

2.1.17.3 Test Activities

Test 1: *The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the*

traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

This activity was performed in conjunction with FCS_TLSC_EXT.2.

Test 2: *The evaluator shall demonstrate that using a certificate without a valid certification path results in an application notification. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the application is notified of the validation failure.*

This activity was performed in conjunction with FCS_TLSC_EXT.2.

2.1.18 Initialization Vector Generation (FCS_IV_EXT.1)

2.1.18.1 TSS Assurance Activity

The evaluator shall examine the key hierarchy section of the TSS to ensure that the encryption of all keys is described and the formation of the IVs for each key encrypted by the same KEK meets FCS_IV_EXT.1.

Windows follows the guidance in [PP MDF] Table 14 when generating initialization vectors as described in [ST] section 6.2.8 SFR Mapping.

2.1.18.2 Guidance Assurance Activities

None Defined.

2.1.18.3 Test Activities

None Defined.

2.1.19 Random Bit Generation (FCS_RBG_EXT.1)

2.1.19.1 TSS Assurance Activity

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix E and the "Clarification to the Entropy Documentation and Assessment Annex".

Microsoft provided CCEVS with entropy documentation as required. The evaluation team provided a review summary in accordance with Appendix E and the "Clarification to the Entropy Documentation and Assessment Annex".

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions described in FCS_RBG_EXT.1.3.

[ST] section 10 Appendix B: Interfaces and Binaries provides the references to the API documentation which includes the security functions (cryptographic algorithms) described in these requirements.

The interfaces CryptographicBuffer.GenerateRandom and CryptographicBuffer.GenerateRandom-Number provide the output of the RBG to applications running on the TSF that request random bits.

2.1.19.2 Guidance Assurance Activities

None Defined.

2.1.19.3 Test Activities

Assurance 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.

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

Implementations Conforming to FIP 140-2 Annex C

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

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

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

Implementations Conforming to NIST Special Publication 800-90A

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the

operational guidance contains appropriate instructions for configuring the RNG functionality.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

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

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

***Entropy input:** the length of the entropy input value must equal the seed length.*

***Nonce:** If a nonce is supported (CTR_DRBG with no 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 <= 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.*

Windows uses algorithm implementations validated under the CAVP. [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP DRBG certificates for Windows 10: 868 (http://csrc.nist.gov/groups/STM/cavp/documents/drbg/drbgnewval.html). The relevant detail is reproduced and highlighted below.

868	**CTR_DRBG**: [Prediction Resistance Tested: Not Enabled; BlockCipher_Use_df: (AES-256) (AES Val#3497)]

[ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies applicable CAVP DRBG certificates for Windows 10 Mobile: 955. The relevant detail is reproduced and highlighted below.

955	**CTR_DRBG**: [Prediction Resistance Tested: Not Enabled; BlockCipher_Use_df: (AES-256) (AES Val#3629)]

2.1.20 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1.1)

2.1.20.1 TSS Assurance Activity

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions (cryptographic algorithms) described in these requirements.

[ST] section 6.2.2 Programming Interfaces provides the API documentation relevant to the security functions (cryptographic algorithms) described in these requirements.

2.1.20.2 Guidance Assurance Activities

None Defined.

2.1.20.3 Test Activities

The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. This application may be used to assist in verifying the cryptographic operation assurance activities for the other algorithm services requirements.

The evaluator used a test app delivered by the developer to request cryptographic operations on the TOE. The evaluator verified that these operations were successful and the results matched the API documentation. Cryptographic operations were requested through APIs identified in [ST]:

- CryptographicEngine.Encrypt
- CryptographicEngine.Decrypt
- HashAlgorithmProvider.HashData[1]
- CryptographicEngine.Sign[2]
- CryptographicEngine.VerifySignature[3]
- KeyDerivationParameters.BuildForPbkdf2
- AsymmetricKeyAlgorithmProvider.CreateKeyPair

[1] Class HashAlgorithmProvider provides method CreateHash that is called by HashData.

[2] Class CryptographicEngine provides methods SignAsync, SignHashedData, and SignhashedDataAsync that calls the same code as Sign.

[3] Class CryptographicEngine provides method VerifySignatureWithHashInput that calls the same code as VerifySignature.

2.1.21 Extended: Cryptographic Algorithm Services (FCS_SRV_EXT.1.2)

2.1.21.1 TSS Assurance Activity

The evaluator shall verify that the API documentation for the secure key storage includes the cryptographic operations by the stored keys.

[ST] section 6.2.2 Programming Interfaces provides the API documentation relevant to the security functions (cryptographic algorithms) described in these requirements. The API's include CryptographicEngine.Sign, CryptographicEngine.Encrypt, and CryptographicEngine.Decrypt.

2.1.21.2 Guidance Assurance Activities

None Defined.

2.1.21.3 Test Activities

The evaluator shall write, or the developer shall provide access to, an application that requests cryptographic operations of stored keys by the TSF. The evaluator shall verify that the results from the operation match the expected results according to the API documentation. The evaluator shall use these APIs to test the functionality of the secure key storage according to the Assurance Activities in FCS_STG_EXT.1.

This activity was performed in conjunction with FCS_SRV_EXT.1.1.

2.1.22 Extended: Cryptographic Key Storage (FCS_STG_EXT.1)

2.1.22.1 TSS Assurance Activity

The assurance activity for this component entails examination of the ST's TSS to determine that the TOE's implements the required secure key storage. The evaluator shall ensure that the TSS contains a description of the key storage mechanism that justifies the selection of "hardware", "hardware-isolated", or "software-based."

[ST] section 6.2.5 Key Storage describes secure key storage. Windows protects keys with a combination of software and hardware mechanisms. The Key Isolation Service provides protected key storage. It relies on Windows process isolation, NTFS discretionary access controls, DPAPI, and BitLocker mechanisms. The section states, "Private keys are protected on disk using DPAPI and BitLocker encryption and access is restricted using the Windows Discretionary Access Control Policy." BitLocker ultimately relies of TPM hardware to protect the REK.

An administrator can perform a wipe on their device to destroy keys while keys for users are protected via DAC and are deleted when a Windows Store Application is deleted.

Additionally, the ST states: "Users and local administrators authorize applications at installation to access shared keys or secrets when an application declares the sharedUserCertificates capability to share

the certificate with other Windows Store Applications for the user. The sharedUserCertificates capability is described further in Restricting Access to System Services."

Section 6.3.1 Restricting Access to System Services, goes into additional information in regards to how restrictions are placed on the Shared User Certificates: "The sharedUserCertificates capability enables a Windows Store Application to access software and hardware certificates, such as certificates stored on a smart card, the certificate is stored in the user's DPAPI profile location instead of the DPAPI profile associated with the Windows Store Application."

2.1.22.2 Guidance Assurance Activities

The evaluator shall review the AGD guidance to determine that it describes the steps needed to import or destroy keys/secrets.

[Mobile Guide] section 13 Managing Certificates covers import and destruction of key and secrets. Subsection 13.1 IT Administrator Guidance describes adding and removing root certificates using an MDM as well as providing links to online guidance. Subsection 13.4 Windows 10 provides the same information for Windows 10 users and local administrators along with instructions for certificate requests. Subsection 13.2 Developer Guidance covers how developers implement key management in applications, which applies when users install applications. Subsection 13.5 Windows 10 Mobile covers deleting keys by wiping the device.

The evaluator shall also verify that the API documentation provided according to Section 6.2.1 includes the security functions (import, use, and destruction) described in these requirements. The API documentation shall include the method by which applications restrict access to their keys/secrets in order to meet FCS_STG_EXT.1.4.

[Mobile Guide] section 13.2.1 Developer Guidance identifies the Windows.Security.Cryptography.Certificates namespace APIs used in key import, for using keys, and for using secrets. The section includes a link to online documentation in an MSDN topic. An application restricts access to keys through choice of certificate enrollment manager class as described in section 13.2.4. Using CertificateEnrollmentManager base class limits access to application that imported or created keys. Using derived class UserCertificateEnrollmentManager limits access by user credentials and account access control lists.

Destruction of keys/secrets is accomplished by wiping the TOE (this functionality is already described as user interfaces – there is no programmatic access to the wipe function).

The [ST] identifies applicable API documentation for Windows 10 and Windows Phone 10 in Section 10 Appendix B: Interfaces.

APIs applicable to key storage are:

1. Import
    a. AsymmetricKeyAlgorithmProvider.ImportKeyPair
        i. Required minimums: Windows 10 device family
    b. CertificateEnrollmentManager.ImportPfxDataAsync
        i. Required minimums: Windows 10 device family
2. Use
    a. CryptographicEngine.Encrypt
        i. Required minimums: Windows 10 device family

b. CryptographicEngine.Decrypt
    i. Required minimums: Windows 10 device family
c. CryptographicEngine.Sign
    i. Required minimums: Windows 10 device family
d. CryptographicEngine.SignAsync
    i. Required minimums: Windows 10 device family
e. CryptographicEngine.SignHashedData
    i. Required minimums: Windows 10 device family
f. CryptographicEngine.SignHashedDataAsync
    i. Required minimums: Windows 10 device family
g. CmsDetachedSignature.GenerateSignatureAsync
    i. Required minimums: Windows 10 device family
h. CmsAttachedSignature.GenerateSignatureAsync
    i. Required minimums: Windows 10 device family
i. Windows.Security.Cryptography.DataProtection
    i. Required minimums: Windows 10 device family

There are no APIs to explicitly destroy keys/secrets, since Windows handles key destruction automatically.

There no APIs for the capability in FCS_STG_EXT.1.4. The capability is determined by sharedUserCertificates as described in [ST] sections 6.2.5 Key Storage and 6.3.1 Restricting Access to System Services. Exceptions are made at application installation.

2.1.22.3 Test Activities

The evaluator shall test the functionality of each security function: ***Test 1**: The evaluator shall import keys/secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that generates a key/secret of each supported type and calls the import functions. The evaluator shall verify that no errors occur during import.*
***Test 2**: The evaluator shall write, or the developer shall provide access to, an application that uses an imported key/secret:* • For RSA, the secret shall be used to sign data. • For ECDSA, the secret shall be used to sign data *In the future additional types will be required to be tested:* • For symmetric algorithms, the secret shall be used to encrypt data. • For persistent secrets, the secret shall be compared to the imported secret. *The evaluator shall repeat this test with the application-imported keys/secrets and a different application's imported keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to use the key/secret imported by the user or by a different application:* • The evaluator shall deny the approvals to verify that the application is not able to use the

key/secret as described.

- The evaluator shall repeat the test, allowing the approvals to verify that the application is able to use the key/secret as described.

If the ST Author has selected "common application developer", this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

Test 3: *The evaluator shall destroy keys/secrets of each supported type according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that destroys an imported key/secret.*

The evaluator shall repeat this test with the application-imported keys/secrets and a different application's imported keys/secrets. The evaluator shall verify that the TOE requires approval before allowing the application to destroy the key/secret imported by the administrator or by a different application:

- The evaluator shall deny the approvals and verify that the application is still able to use the key/secret as described.
- The evaluator shall repeat the test, allowing the approvals and verifying that the application is no longer able to use the key/secret as described.

If the ST Author has selected "common application developer", this test is performed by either using applications from different developers or appropriately (according to API documentation) not authorizing sharing.

The evaluator used a suite of test apps to create and approve cryptographic keys, use the keys and allow other applications to use the keys. The evaluator verified that all these operations were successful. The evaluator then destroyed the keys and verified that the apps could no longer use them.

2.1.23 Extended: Encrypted Cryptographic Key Storage (FCS_STG_EXT.2)

2.1.23.1 TSS Assurance Activity

The evaluator shall review the TSS to determine that the TSS includes key hierarchy description of the protection of each DEK for data-at-rest, of software-based key storage, of long-term trusted channel keys, and of KEK related to the protection of the DEKs, long-term trusted channel keys, and software-based key storage. This description must include a diagram illustrating the key hierarchy implemented by the TOE in order to demonstrate that the implementation meets FCS_STG_EXT.2. The description shall indicate how the functionality described by FCS_RBG_EXT.1 is invoked to generate DEKs (FCS_CKM_EXT.2), the key size (FCS_CKM_EXT.2 and FCS_CKM_EXT.3) for each key, how each KEK is formed (generated, derived, or combined according to FCS_CKM_EXT.3), the integrity protection method for each encrypted key (FCS_STG_EXT.3), and the IV generation for each key encrypted by the same KEK (FCS_IV_EXT.1). More detail for each task follows the corresponding requirement.

[ST] describes the key hierarchy in sections 6.2.4 Encrypting the Device with BitLocker, 6.2.5 Key Storage, and 6.2.6 Protecting Data with DPAPI describe how Windows uses key and data encryption.. See above section 2.1.7.1 above in Cryptographic Key Random Generation (FCS_CKM_EXT.2) for a

summary of the key hierarchy. See 2.1.18.1 above in Initialization Vector Generation (FCS_IV_EXT.1) for IV generation by cipher modes.

The evaluator shall examine the key hierarchy section of the TSS to ensure that each key (DEKs, software-based key storage, and KEKs) is encrypted by keys of equal or greater security strength using one of the selected modes.

See section 2.1.7.1 above in Cryptographic Key Random Generation (FCS_CKM_EXT.2) for a summary of the key hierarchy. See above 2.1.8.1 above in Cryptographic Key Generation Extended (FCS_CKM_EXT.3) for KEK key sizes.

See 2.1.8.1 above in Cryptographic Key Generation Extended (FCS_CKM_EXT.3) regarding strength of KEK and chains of keys.

The evaluator shall examine the key hierarchy description in the TSS section to verify that each DEK and software-stored key is encrypted according to FCS_STG_EXT.2.

See 2.1.7.1 above in Cryptographic Key Random Generation (FCS_CKM_EXT.2) for a summary of the key hierarchy. See 2.1.8.1 Cryptographic Key Generation Extended (FCS_CKM_EXT.3) for KEK key sizes. See 2.1.18.1 above in Initialization Vector Generation (FCS_IV_EXT.1) for cipher modes.

2.1.23.2 Guidance Assurance Activities

None defined.

2.1.23.3 Test Activities

None defined.

2.1.24 Extended: Integrity of encrypted key storage (FCS_STG_EXT.3)

2.1.24.1 TSS Assurance Activity

The evaluator shall examine the key hierarchy description in the TSS section to verify that each encrypted key is integrity protected according to one of the options in FCS_STG_EXT.3.

The Table 2 below summarizes the information from the key hierarchy. For each key, the table identifies the key type and the mechanism that provides integrity of the key.

Table 2 Key Integrity Summary

Key	Type	Integrity
Storage Primary Seed	REK	N/A hardware-isolated not encrypted
Storage Root Key	KEK	N/A when derived from SPS Keyed hash when in Protected Storage (as per [TPM 2.0 Arch] section 22 Protected Storage)

Intermediate keys	KEK	CCM in accordance with [PP MDF] Table 14
VMK	KEK	CCM in accordance with [PP MDF] Table 14
DPAPI password-based encryption key	KEK	N/A PBKDF
DPAPI Master Secret	KEK	Keyed hash
Device User Credential Keys	KEK	CCM in accordance with [PP MDF] Table 14
User's public/private key pairs	KEK	Microsoft proprietary

2.1.24.2 Guidance Assurance Activities

None defined.

2.1.24.3 Test Activities

None defined.

2.1.25 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.1)

2.1.25.1 TSS Assurance Activity

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified.

[ST] section 6.2.7.1.1 TLS and EAP TLS lists the cipher suites supported by Windows.

The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

[ST] section 6.2.7.1.1 TLS and EAP TLS distinguishes between the cipher suites supported by Windows 10 and the cipher suites supported by Windows 10 Mobile, which match FCS_TLSC_EXT.1(C) and FCS_TLSC_EXT.1(M), respectively.

2.1.25.2 Guidance Assurance Activities

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

[Mobile Guide] section 5 Managing TLS lists the cipher suites the TOE supports. The section covers configuring TLS via MDM[4] and for Windows 10 as a user and local administrator. The section includes links for Windows 10 to guidance to configure a server to allow only the specified cipher suites.

Section 6 provides the correspondence between cipher suites names used in the security target and the cipher suite names used in TOE configuration:

ST Name: TLS_RSA_WITH_AES_128_CBC_SHA
Config Name: TLS_RSA_WITH_AES_128_CBC_SHA

ST name: TLS_RSA_WITH_AES_256_CBC_SHA
Config Name: TLS_RSA_WITH_AES_256_CBC_SHA

ST Name: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
Config Name: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P256

and/or

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA_P384

Note: RFC 4492 permits the use of multiple curves for the ECDHE-ECDSA key exchange. RFC 4492 Section 5.1 states: A server participating in an ECDHE-ECDSA key exchange may use different curves for (i) the ECDSA key in its certificate, and (ii) the ephemeral ECDH key in the ServerKeyExchange message.

ST Name: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
Config Name: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P256

and/or

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA_P384

Note: RFC 4492 permits the use of multiple curves for the ECDHE-ECDSA key exchange. RFC 4492 Section 5.1 states: A server participating in an ECDHE-ECDSA key exchange may use different curves for (i) the ECDSA key in its certificate, and (ii) the ephemeral ECDH key in the ServerKeyExchange message.

ST Name: TLS_RSA_WITH_AES_128_CBC_SHA256
Config Name: TLS_RSA_WITH_AES_128_CBC_SHA256

ST Name: TLS_RSA_WITH_AES_256_CBC_SHA256
Config Name: TLS_RSA_WITH_AES_256_CBC_SHA256

[4] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

ST Name: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
Config Name: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256

ST Name: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
Config Name: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384

ST Name: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
Config Name: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256

ST Name: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
Config Name: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384

2.1.25.3 Test Activities

The evaluator shall also perform the following tests:

Test 1: *The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).*

The evaluator successfully negotiated an EAP-TLS connection from the TOE using each of the claimed cipher suites in the Security Target.

Test 2: *The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

Test 3: *The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.*

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

Test 4: *The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.*

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

Test 5: *The evaluator shall perform the following modifications to the traffic:*

- Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
-
- Last bullet in Test 5 was modified per TD0034
- Send a valid Server Finished message in plaintext and verify the client sends a fatal alert upon receipt and does not send any application data. The server's finished message shall contain valid verify_data and shall parse correctly using a network protocol analysis tool.

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

2.1.26 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.2)

2.1.26.1 TSS Assurance Activity

None defined.

2.1.26.2 Guidance Assurance Activities

The evaluator shall check that the AGD guidance contains instructions for the administrator to configure the list of Certificate Authorities that are allowed to sign certificates or to configure the FQDN of the authentication server certificate that will be accepted by the TOE in the EAP-TLS exchange.

[Mobile Guide] section 5 Managing EAP-TLS contains the guidance to manage the EAP-TLS exchange. The section identifies Wi-Fi profiles as the mechanisms for MDM configuration[5]. For Windows 10, the section includes links to EAP settings and certificate management for local administrators. The EAP settings topic contains configuration information specific to the EAP-TLS authentication method. In particular, it covers configuration of certificate validation for network connections based on EAP-TLS. [Mobile Guide] Section 13 Certificate Management contains the guidance to configure the list of Certificate Authorities that are allowed to sign certificates.

2.1.26.3 Test Activities

The evaluator shall also perform the following tests:

Test 1: *Following the guidance provided by the AGD guidance, a CA or an FQDN will be configured as "acceptable" for authentication server certificates and then the evaluator will start a wireless connection and verify that the wireless client is able to successfully connect. The evaluator will then configure the system such that an otherwise valid certificate is signed by a CA that is not allowed by the TOE or presents a FQDN that is not allowed by the TOE. Attempts to authenticate to an authentication server presenting such a certificate should result in the connection being refused. If the TOE supports both methods of limiting the acceptable authentication servers, the evaluator shall repeat this test twice, once with each method.*

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. The activity in FCS_TLSC_EXT.2 is an expansion of this activity that tests the values of the FQDN by using several modified values of the CN and SAN that test both the invalid and valid cases.

2.1.27 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.3)

2.1.27.1 TSS Assurance Activity

None defined.

[5] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

2.1.27.2 Guidance Assurance Activities

None defined.

2.1.27.3 Test Activities

The evaluator shall also perform the following tests:

Test 1: *The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.*

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

2.1.28 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.4)

2.1.28.1 TSS Assurance Activity

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

[ST] section 6.2.8 SFR Mapping identifies mutual authentication for TLS 1.0, 1.1 and 1.2. The cipher suites specified in FCS_TLSC_EXT.1.1 all require client-side certificate for mutual authentication.

2.1.28.2 Guidance Assurance Activities

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

[Mobile Guide] section 5 Managing TLS states that no configuration is necessary to use client authentication on the device once a device has client authentication certificates. Section 13 Managing Certificates contains information on configuring a device to enroll for client certificates.

2.1.28.3 Test Activities

The evaluator shall also perform the following tests:

Test 1: *The evaluator shall perform the following modification to the traffic:*

- Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify the connection is

unsuccessful.

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

2.1.29 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.5)

2.1.29.1 TSS Assurance Activity

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured.

[ST] section 6.2.7.1.1 TLS and EAP TLS states that when negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client Hello message both its supported elliptic curves extension (i.e., secp256r1, secp384r1, and secp521r1) as well as signature algorithm (i.e., SHA256, SHA384, and SHA512).

2.1.29.2 Guidance Assurance Activities

If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

[ST] section 6.2.7.1.1 TLS and EAP TLS states that when negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client Hello message its supported elliptic curves extension.

2.1.29.3 Test Activities

The evaluator shall also perform the following test:

Test: *The evaluator shall configure the server to perform an ECDHE key exchange message in the TLS connection using a non-supported ECDHE curve (for example, P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.*

The evaluator used TLS server utility to force the use of the P-192 curve when the TOE negotiates the P-256 curve. The evaluator verified that the TOE rejects this curve and ceases the connection.

2.1.30 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.6)

2.1.30.1 TSS Assurance Activity

The evaluator shall verify that TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured.

[ST] section 6.2.7.1.1 TLS and EAP TLS states that when negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client Hello message both its supported elliptic

curves extension, i.e., secp256r1, secp384r1, and secp521r1 as well as signature algorithm, i.e., SHA256, SHA384, and SHA512.

[ST] section 6.2.7.1.1 states that each Windows component that uses TLS checks that the identifying information in the certificate matches what is expected, the component should reject the connection, these checks include checking the expected Distinguished Name (DN), Subject Name (SN), or Subject Alternative Name (SAN) attributes along with the applicable extended key usages. The DN, and any Subject Alternative Name, in the certificate is checked against the identity of the remote computer's DNS entry or IP address to ensure that it matches. Matching criteria is further described at http://technet.microsoft.com/en-us/library/cc783349(v=WS.10).aspx, see the "Server Certificate Message" section.

2.1.30.2 Guidance Assurance Activities

If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.

As indicated in section 2.1.29.1 above, the [ST] section 6.2.7.1.1 TLS and EAP TLS states that when negotiating a TLS 1.2 elliptic curve cipher suite, Windows will include automatically as part of the Client Hello message its signature algorithm, i.e., SHA256, SHA384, and SHA512.

Section 6.2.1 Local Administrator Guidance of the [Mobile Guide] states that the signature algorithm is not configurable in Windows 10 for TLS.

2.1.30.3 Test Activities

The evaluator shall also perform the following test:

The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

2.1.31 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.7)

2.1.31.1 TSS Assurance Activity

None defined.

2.1.31.2 Guidance Assurance Activities

None defined.

2.1.31.3 Test Activities

None defined.

2.1.32 Extended: EAP TLS Protocol (FCS_TLSC_EXT.1.8)

2.1.32.1 TSS Assurance Activity

None defined.

2.1.32.2 Guidance Assurance Activities

None defined.

2.1.32.3 Test Activities

The evaluator shall perform the following tests: *Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the "renegotiation_info" field or the SCSV ciphersuite is included in the ClientHello packet during the initial handshake.*
Test 2: *The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.*
Test 3: *The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client_verify_data" or "server_verify_data" value and verify that the client terminates the connection.*

Because of the similarity in assurance activities, this activity was performed in conjunction with FCS_TLSC_EXT.2. This wording of this activity and the one explained in FCS_TLSC_EXT.2 are exactly the same.

2.1.33 Extended: TLS Protocol (FCS_TLSC_EXT.2.1)

2.1.33.1 TSS Assurance Activity

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified.

[ST] section 6.2.7.1.1 TLS and EAP TLS lists the cipher suites supported by SChannel, which match FCS_TLS_EXT.2.1.

The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

[ST] section 6.2.7.1.1 TLS and EAP TLS lists the cipher suites supported by SChannel, which match FCS_TLS_EXT.2.1.

2.1.33.2 Guidance Assurance Activities

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

See AAR Sections 2.1.25.2 and 2.1.26.2 Guidance Assurance Activities for analysis.

2.1.33.3 Test Activities

The evaluator shall write, or the ST author shall provide, an application for the purposes of testing TLS. The evaluator shall also perform the following tests:

Test 1: *The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).*

The evaluator connected the TOE to a web server (via HTTPS) to negotiate each of the cipher suites claimed in the Security Target. The evaluator confirmed that each negotiation succeeded and used the correct cipher suite.

Test 2: *The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*

The evaluator attempted to establish a TLS connection between the TOE and a server with the server providing a certificate that did not contain the Server Authentication purpose. The evaluator verified that the TOE rejected the connection.

Test 3*: The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA certificate while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.*

The evaluator attempted to establish a TLS connection between the TOE and a server with an ECDSA server certificate when a RSA cipher suite was negotiated. The evaluator verified that upon receipt of the certificate the TOE rejects the connection.

Test 4: *The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.*

The evaluator attempted to establish a TLS connection between the TOE and a server using the TLS_NULL_WITH_NULL_NULL cipher suite. The evaluator verified that TOE rejected this connection.

Test 5: *The evaluator shall perform the following modifications to the traffic:*

- Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
- Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (ifusing a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.
- Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.
- (conditional) If a ECDHE or DHE ciphersuite is selected, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
- Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
-
- Last bullet in Test 5 was modified per TD0034
- Send a valid Server Finished message in plaintext and verify the client sends a fatal alert upon receipt and does not send any application data. The server's finished message shall contain valid verify_data and shall parse correctly using a network protocol analysis tool.

The evaluator attempted a TLS connection between the TOE and a server using the modifications to the traffic bulleted above. For each of these modifications, the evaluator confirmed that the TOE rejected the connection.

2.1.34 Extended: TLS Protocol (FCS_TLSC_EXT.2.2)

2.1.34.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

[ST] section 6.2.7.1.1 TLS and EAP TLS covers the DN check (search "checking the expected Distinguished Name (DN), Subject Name (SN), or Subject Alternative Name (SAN)").

The DN and SAN, as explained in the ST, is as follows: 'The DN, and any Subject Alternative Name, in the certificate, is checked against the identity of the remote computer's DNS entry or IP address to ensure that it matches as described at http://technet.microsoft.com/en-us/library/cc783349(v=WS.10).aspx, and in particular the "Server Certificate Message" section.'

A certificate that uses a wildcard in the leftmost portion of the resource identifier (i.e., *.contoso.com) can be accepted for authentication, otherwise the certificate will be deemed invalid.

The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

[ST] section 6.2.7.1.1 TLS and EAP TLS states that Windows does not provide a general-purpose capability to "pin" TLS certificates.

2.1.34.2 Guidance Assurance Activities

The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS. In particular, the AGD guidance should describe the API used by applications for configuring the reference identifier.

[Mobile Guide] sections 5.3.1 Local Administrator Guidance and 5.4 Windows 10 Mobile indicate that the reference identifier in Windows for TLS is the URL of the server; and that no configuration of the reference identifier is required.

2.1.34.3 Test Activities

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

***Test 1:** The evaluator shall present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator shall verify that the connection fails.*

***Test 2:** The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat*

this test for each supported SAN type.
Test 3: *The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.*
Test 4: *The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.*
Test 5: *The evaluator shall perform the following wildcard tests with each supported type of reference identifier:* • The evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails. • The evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails. • The evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.
Test 6: [conditional] *If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.*
Test 7: [conditional] *If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.*

The evaluator created a server certificate with each of the modifications above. The evaluator then attempted a TLS connection between the TOE and the server using each of the modified certificates. The evaluator verified that when the connection was expected to succeed it did and when it should fail, the TOE rejected the connection.

2.1.35 Extended: TLS Protocol (FCS_TLSC_EXT.2.3)

2.1.35.1 TSS Assurance Activity

None defined.

2.1.35.2 Guidance Assurance Activities

None defined.

2.1.35.3 Test Activities

The evaluator shall perform the following test:

Test 1: *The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.*

The evaluator loaded a server certificate onto the sever that did not chain to a trusted root authority. The evaluator the attempted a connection between the TOE and server and verified that upon receipt of the certificate the TOE rejected the connection.

As specified in [ST] section 6.4.2 X.509 Certificate Validation and Generation, the evaluator observed if certificate validation fails, Windows will not establish a trusted network channel except in the case of HTTPS web browsing. Windows informs the user and seeks their consent before establishing a HTTPS web browsing session, which is the behavior specified in FCS_HTTPS_EXT.1.3.

2.1.36 Extended: TLS Protocol (FCS_TLSC_EXT.2.4)

2.1.36.1 TSS Assurance Activity

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

[ST] section 6.2.8 SFR Mapping identifies mutual authentication for TLS 1.0, 1.1 and 1.2. The cipher suites specified in FCS_TLSC_EXT.2.1 all require client-side certificate for mutual authentication.

2.1.36.2 Guidance Assurance Activities

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

[Mobile Guide] section 6.1 Managing TLS states that no configuration is necessary to use client authentication on the device once a device has client authentication certificates. Section 13 Managing Certificates contains information on configuring a device to enroll for client certificates.

2.1.36.3 Test Activities

The evaluator shall perform the following test:

Test 1: The evaluator shall perform the following modification to the traffic:

- Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify the connection is unsuccessful.

The evaluator configured the TOE to connect to the TLS server using mutual authentication and attempted a connection. The evaluator then modified a byte in the CA field in the Server's Certificate Request message and verified that the TOE rejected the connection.

2.1.37 Extended: TLS Protocol (FCS_TLSC_EXT.2.5)

2.1.37.1 TSS Assurance Activity

The evaluator shall verify that TSS describes the Supported Elliptic Curves Extension and whether the required behavior is performed by default or may be configured.

[ST] section 6.2.7.1.1 TLS and EAP TLS states that when negotiating a TLS 1.2 elliptic curve cipher suite, Windows will automatically include as part of the Client Hello message both its supported elliptic curves extension (i.e., secp256r1, secp384r1, and secp521r1) as well as signature algorithm (i.e., SHA256, SHA384, and SHA512).

2.1.37.2 Guidance Assurance Activities

If the TSS indicates that the Supported Elliptic Curves Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves Extension.

See AAR section 2.1.29.2 above.

2.1.37.3 Test Activities

Testing for this element are performed in conjunction with the assurance activities for FPT_TST_EXT.2.1.

This activity was performed in conjunction with FCS_TLSC_EXT.1.5.

2.1.38 Extended: TLS Protocol (FCS_TLSC_EXT.2.6)

2.1.38.1 TSS Assurance Activity

The evaluator shall verify that TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured.

[ST] section 6.2.7.1.1 TLS and EAP TLS states that when negotiating a TLS 1.2 elliptic curve cipher suite, Windows will automatically include as part of the Client Hello message both its supported elliptic curves extension, i.e., secp256r1, secp384r1, and secp521r1 as well as signature algorithm, i.e., SHA256, SHA384, and SHA512.

[ST] section 6.2.7.1.1 states that each Windows component that uses TLS checks that the identifying information in the certificate matches what is expected, the component should reject the connection, these checks include checking the expected Distinguished Name (DN), Subject Name (SN), or Subject Alternative Name (SAN) attributes along with the applicable extended key usages. The DN, and any Subject Alternative Name, in the certificate is checked against the identity of the remote computer's DNS entry or IP address to ensure that it matches. Matching criteria is further described at http://technet.microsoft.com/en-us/library/cc783349(v=WS.10).aspx, see the "Server Certificate Message" section.

2.1.38.2 Guidance Assurance Activities

If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension.

See AAR Section 2.1.30.2 above for analysis.

2.1.38.3 Test Activities

The evaluator shall also perform the following test:

Test: The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

The evaluator attempted a TLS connection between the TOE and a server. The evaluator configured the server to use MD2 hash algorithm and verified that the TOE rejected the connection.

2.1.39 Extended: TLS Protocol (FCS_TLSC_EXT.2.7)

2.1.39.1 TSS Assurance Activity

None defined.

2.1.39.2 Guidance Assurance Activities

None defined.

2.1.39.3 Test Activities

None defined.

2.1.40 Extended: TLS Protocol (FCS_TLSC_EXT.2.8)

2.1.40.1 TSS Assurance Activity

None defined.

2.1.40.2 Guidance Assurance Activities

None defined.

2.1.40.3 Test Activities

The evaluator shall perform the following tests:

Test 1: *The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the "renegotiation_info" field or the SCSV ciphersuite is included in the ClientHello packet during the initial handshake.*

The evaluator observed a TLS connection between the TOE and sever and verified that the renegotiation_info field was contained in the ClientHello packet.

Test 2: *The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.*

The evaluator configured the TOE to connect to a TLS server and modified the renegotiation_info length in the ServerHello packet. The evaluator verified that the TOE rejected the connection.

Test 3: *The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client verify data" or "server verify data" value and verify that the client terminates the connection.*

The evaluator modified the server_verify_data value during a connection attempt between the TOE and the TLS server. The evaluator confirms that the TOE rejected the connection.

2.2 User Data Protection (FDP)

2.2.1 Extended: Security Access Control (FDP_ACF_EXT.1.1)

2.2.1.1 TSS Assurance Activity

The evaluator shall ensure the TSS lists all system services available for use by an application. The

evaluator shall also ensure that the TSS describes how applications interface with these system services, and means by which these system services are protected by the TSF.

[ST] section 6.3.1 Restricting Access to System Services lists device resources accessible to Windows Store Apps. Section 6.3.1 describes the package manifest for Windows Store Apps (search "package manifest for the application") for interfacing with system services. The section describes the Windows App Container that mediates access to system services (search "Windows App Container").

The TSS shall describe which of the following categories each system service falls in:

1) No applications are allowed access

2) Privileged applications are allowed access

3) Applications are allowed access by user authorization

4) All applications are allowed access

For all applications, Windows requires user authorization for each system services at application installation (search "user is prompted to authorize" in [ST] section 6.3.1 Restricting access to System Services).

Privileged applications include any applications developed by the TSF developer. The TSS shall describe how privileges are granted to third-party applications. For both types of privileged applications, the TSS shall describe how and when the privileges are verified and how the TSF prevents unprivileged applications from accessing those services.

Section 6.3.1 Restricting Access to System Services identifies package manifest, capability, and the Windows App Container as the mechanisms for granting privilege to application. If a capability for a system service is included in an application's package manifest and a user authorizes that capability when installing the application, then the Windows App Container will grant the application access to the service (search "managed by a capability").

Microsoft distinguishes capabilities through the publication process. Individual developers registered with Windows Store accounts can include in application package manifests the capabilities listed in [ST] Table 17 General Use Capabilities and Table 18 Device Capabilities. Microsoft provides additional review for applications that include capabilities in Table 19 Special Use Capabilities as well as requiring the developers have a registered company account with the Windows Store (search "additional capabilities").

For any services for which the user may grant access, the evaluator shall ensure that the TSS identifies whether the user is prompted for authorization when the application is installed, or during runtime.

For all applications, Windows requires user authorization for each system services at application installation (search "user is prompted to authorize" in [ST] section 6.3.1 Restricting Access to System Services).

2.2.1.2 Guidance Assurance Activities

The evaluator shall ensure that the operational user guidance contains instructions for restricting application access to system services.

Application access to system services is determined at installation as described above. [Mobile Guide] section 6 Managing Apps covers restricting ability to install, run, and remove applications. Section 6.1 IT Administrator Guidance identifies the capability of MDM[6] solutions to install, remove, and restrict the ability to run for applications. Section 6.2.1 provides guidance to Windows 10 local administrators for restricting user ability to install and run applications. Section 6.2.1 includes instructions for installing and removing applications. As noted in [ST] section 6.3.1 Restricting Access to System Services a user authorizes application access to system services at installation not through configuration.

2.2.1.3 Test Activities

Assurance Activity Note: The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.

The evaluator shall write, or the developer shall provide, applications for the purposes of the following tests.

Test 1: *For each system service to which no applications are allowed access, the evaluator shall attempt to access the system service with a test application and verify that the application is not able to access that system service.*

Not Applicable to the TOE: There are no system services to which no applications are allowed access.

Test 2: *For each system service to which only privileged applications are allowed access, the evaluator shall attempt to access the system service with an unprivileged application and verify that the application is not able to access that system service. The evaluator shall attempt to access the system service with a privileged application and verify that the application can access the service.*

Not Applicable to the TOE: There are no system services to which only privileged (developed and included in the TSF by Microsoft) applications are allowed access.

Test 3: *For each system service to which the user may grant access, the evaluator shall attempt to access the system service with a test application. The evaluator shall ensure that either the system blocks such accesses or prompts for user authorization. The prompt for user authorization may occur at runtime or at installation time, and should be consistent with the behavior described in the TSS.*

To test this requirement the developer provided an app test suite known as the "SysUse" apps. Each app tests a specific capability and collectively the apps fulfill the requirement.

Test 4: *For each system service listed in the TSS that is accessible by all applications, the evaluator shall test that an application can access that system service.*

All system services are available to all applications; therefore this test is satisfied by Test 3 above.

[6] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

2.2.2 Extended: Security Access Control (FDP_ACF_EXT.1.2)

2.2.2.1 TSS Assurance Activity

The evaluator shall examine the TSS to verify that it describes which data sharing is permitted between applications, which data sharing is not permitted, and how disallowed sharing is prevented.

Section 6.6.1 states Application Containers ("App Containers") provide an execution environment for Universal Windows Applications. The environment prevents Universal Windows Applications from accessing data created by other Universal Windows Applications (that is, private data in [PP MDF] terminology) except through brokered operating system services such as the File Picker dialog. Table 17 General Use Capabilities lists shared data resources and the capability required to access each one.

By definition, Universal Windows Applications do not have the capability to launch ("execute" in the language of the [PP MDF]) other programs. An application can read or write to a file. Table 19 Special Use Capabilities lists the Documents capability for access to the documents library.

2.2.2.2 Guidance Assurance Activities

None defined.

2.2.2.3 Test Activities

Test: *The evaluator shall write, or the developer shall provide, two applications, one which saves data containing a unique string and the other which attempts to access that data. If "groups of applications" is selected, the applications shall be placed into different groups. If "private data" is selected, the application shall not write to a designated shared storage area. The evaluator shall verify that the second application is unable to access the stored unique string. The evaluator shall grant access, either as a user, the administrator, or by using a third application with a common application developer to the first, and verify that the application is able to access the stored unique string.*

The developer provided two apps, ReadAppData and WriteAppData. The evaluator used WriteAppData to create a file that ReadAppData did not have access to. The evaluator verified that ReadAppData was not able to access the file; then the evaluator granted access to the file and the evaluator verified ReadAppData was allowed access.

2.2.3 Extended: Security Access Control (FDP_ACF_EXT.1.3)

2.2.3.1 TSS Assurance Activity

None defined.

2.2.3.2 Guidance Assurance Activities

None defined.

2.2.3.3 Test Activities

Assurance 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 consumer Mobile Device products.*

Test 1: *The evaluator shall write, or the developer shall provide, an application which attempts to store a file with both write and execute permissions. The evaluator shall verify that this action fails and that the permissions on the file are not simultaneously write and execute.*

The Windows class Windows.Storage.FileAccessMode does not offer a method to attempt to store a file with both write and execute permissions. See: https://msdn.microsoft.com/en-us/library/windows/apps/windows.storage.fileaccessmode.aspx.

Test 2: *The evaluator shall traverse the file system examining the permission on each TSF file to verify that no file has both write and execute permissions set.*

This activity was performed in conjunction with FPT_AEX_EXT.4

2.2.4 Extended: Limitation of Bluetooth Device Access (FDP_BLT_EXT.1)

2.2.4.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes the mechanism used to prevent unrestricted access to paired Bluetooth devices (and/or their communication data) by every application with access to the Bluetooth system service on the TOE (as listed in FDP_ACF_EXT.1). The evaluator shall verify that this method either restricts access to a single application or provides explicit control of the applications that may communicate with the paired Bluetooth device.

Windows uses the device capabilities described in section 6.3.1 Restricting Access to System Services to restrict access to Bluetooth devices. The Bluetooth GATT and Bluetooth RFCOMM device capabilities allow the Windows Store App to access Bluetooth services. The Bluetooth device capability allows applications to communicate with already paired Bluetooth devices.

2.2.4.2 Guidance Assurance Activities

None defined.

2.2.4.3 Test Activities

None defined.

2.2.5 Extended: Protected Data Encryption (FDP_DAR_EXT.1)

FDP_DAR_EXT.1(128) and FDP_DAR_EXT.1(256) correspond to FDP_DAR_EXT.1 in the [PP MDF] protection profile.

2.2.5.1 TSS Assurance Activity

The evaluator shall verify that the TSS section of the ST indicates which data is protected by the DAR implementation and what data is considered TSF data. The evaluator shall ensure that this data includes all protected data.

All user data and all Windows data are encrypted on the device. [ST] section 6.3.2 Data at Rest Protection describes encryption of the entire storage volume as protected by BitLocker full disk encryption, this includes user data, Windows configuration (TSF) data, and all programs other than the BitLocker programs needed to unlock the drive.

2.2.5.2 Guidance Assurance Activities

The evaluator shall review the AGD guidance to determine that the description of the configuration and use of the DAR protection does not require the user to perform any actions beyond configuration and providing the authentication credential.

[Mobile Guide] section 7 Managing Volume Encryption covers data at rest protection. An administrator configures volume encryption either through an MDM[7] solution or as a local Windows 10 administrator. A Windows 10 Mobile user may enable and disable volume encryption as described in section 7.3.1 BitLocker and Device Encryption do not require the user to perform any actions beyond configuration and providing the authentication credential.

The evaluator shall also review the AGD guidance to determine that the configuration does not require the user to identify encryption on a per-file basis.

The encryption is configured on the entire operation system volume or removable volumes. The configuration does not require the user to identify encryption on a per-file basis.

2.2.5.3 Test Activities

Assurance Activity Note: *The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

Test 1: *The evaluator shall enable encryption according to the AGD guidance. The evaluator shall create user data (non-system) either by creating a file or by using an application. The evaluator shall use a tool provided by the developer to verify that this data is encrypted when the product is powered off, in conjunction with Test 1 for FIA_UAU_EXT.1.*

[7] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

This activity was performed in conjunction with FIA_UAU_EXT.1.

2.2.6 Extended: Subset information flow control (FDP_IFC_EXT.1)

2.2.6.1 TSS Assurance Activity

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

[ST] section 6.3.4 VPN Client describes the native Windows IPsec VPN client and identifies the Windows Networking VPN APIs and device capabilities a developer could use to implement a VPN client. The evaluation team used the native Windows 10 IPsec VPN client in the evaluation. [ST] does not claim IPsec in FTP_ITC_EXT.1 and Microsoft has not yet evaluated the native VPN client against the *Protection Profile for IPsec Virtual Private Network (VPN) Clients*.

[ST] section 6.3.4 describes routing IP traffic through processes (search "The components responsible for routing IP traffic"). Section 6.3.4 states that all traffic is routed through the IPsec tunnel except three items as listed (search "routed through the IPsec tunnel except"). Windows routes all IP traffic through an enabled VPN client, except as noted (search "all IP traffic is routed").

The evaluator shall verify that the TSS section describes any differences in the routing of IP traffic when using any supported baseband protocols (e.g. WiFi or, LTE).

[ST] section 6.3.4 VPN Client does not identify any differences in the routing of Wi-Fi. The IPsec VPN is an end-to-end internetworking technology and so VPN sessions can be established over physical network protocols such as wireless LAN (Wi-Fi) or local area network.

The evaluator shall verify that one (or more) of the following options is addressed by the documentation:

- *The description above indicates that if a VPN client is enabled, all configurations route all Data Plane traffic through the tunnel interface established by the VPN client.*
- *The AGD guidance describes how the user and/or administrator can configure the TSF to meet this requirement.*
- *The API documentation includes a security function that allows a VPN client to specify this routing.*

[Mobile Guide] Section 8 is Managing VPN. Subsection 8.1 *IT Administrator's Guidance* describes that an MDM system may be used to administer VPN profiles. The Windows IPsec VPN client can be configured by the MDM IT administrator, when the device is enrolled. The evaluated configuration requires that all network traffic other than traffic necessary to establish the VPN connection go through the VPN tunnel. This is done by verifying that the VPN configuration pushed down by the MDM is configured to "Send all traffic through the VPN connection".

[Mobile Guide] subsection 8.2 *Windows 10* provides a TechNet topic that describes how to create a VPN connection. The Add-VpnConnection and Set-VpnConnection topic cover configuration to prevent split tunneling.

2.2.6.2 Guidance Assurance Activities

See TSS Assurance Activity Above

2.2.6.3 Test Activities

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

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

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

Step 3 *- The evaluator shall examine the traffic from both step one and step two to verify that all Data Plane traffic is encapsulated by IPsec. The evaluator shall examine the Security Parameter Index (SPI) value present in the encapsulated packets captured in Step two from the TOE to the Gateway and shall verify this value is the same for all actions used to generate traffic through the VPN. Note that it is expected that the SPI value for packets from the Gateway to the TOE is different than the SPI value for packets from the TOE to the Gateway. The evaluator shall be aware that IP traffic on the cellular baseband outside of the IPsec tunnel may be emanating from the baseband processor and shall verify with the manufacturer that any identified traffic is not emanating from the application processor.*

Step 4 *- The evaluator shall perform an ICMP echo from the TOE to the IP address of another device on the local wireless network and shall verify that no packets are sent using the sniffing tool. The evaluator shall attempt to send packets to the TOE outside the VPN tunnel (i.e. not through the VPN gateway), including from the local wireless network, and shall verify that the TOE discards them.*

The evaluator connected the TOE to an IPsec VPN and verified traffic was encapsulated by IPsec when connected to the VPN and traffic was plaintext when not connected to the VPN. The evaluator also

verified the SPI values were the same for all packets going to the TOE from the Gateway and from the TOE to the Gateway. The evaluator configured the TOE to not allow traffic to or from outside the VPN when connected to the VPN and verified that any attempt to bypass the VPN was denied by the TOE or dropped by the TOE.

2.2.7 Extended: User Data Storage (FDP_STG_EXT.1)

2.2.7.1 TSS Assurance Activity

The evaluator shall ensure the TSS describes the Trust Anchor Database implemented that contain certificates used to meet the requirements of this PP. This description shall contain information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access (for example, unix permissions) in accordance with the permissions established in FMT_SMF_EXT.1 and FMT_MOF_EXT.1.1.

[ST] section 6.3.3 Certificate Storage explains Windows stores trusted root certificates in certificates stores, which serve as the Trust Anchor Database. The Trust Anchor Database consists of multiple Trusted Root Certificate stores. The Trust Anchor Database consists of one Trusted Root Certificate store for each user account, and a Trusted Root Certificate store for the computer account. Access to a certificate store is managed by the discretionary access control policy in Windows such that only the authorized administrator, i.e., the user or the local administrator, can add or remove entries. Certificates which are used by applications, for example, IPsec and TLS, are also placed in certificate stores for the user. In addition to the standard certificate revocation processes, application certificates can be loaded by either using administrative tools such as certutil.exe, changes to the trusted root certificates can be made using Certificate Trust Lists (https://msdn.microsoft.com/en-us/library/windows/desktop/aa376545(v=vs.85).aspx) .

[ST] Section 6.5 Security Management describes the authorized administrator function (search “Trust Anchor Database” in Section 6.5).

Windows implements a more robust Trust Anchor Database and a more granular administrator role than are described in the protection profile. FMT_SMF_EXT.1 specifies two capabilities related to the Trust Anchor Database:

13. import X.509v3 certificates into the Trust Anchor Database
14. remove imported X.509v3 certificates and [[all X.509v3 certificates]] in the Trust Anchor Database,

FMT_SMF_EXT.1 capability 13 is offered to administrators and restricted to administrators. Capability 14 is restricted to standard users for Windows 10 and to administrators for enrolled Windows 10 Mobile devices. This division of capabilities is shown in Table 20 Mobile Device Management Capabilities. See rows containing "Trust Anchor Database".

In summary, only an administrator can add certificates to the Trust Anchor Database as follows:

- A user acting as administrator for the user's Trusted Root Certificate store
- A Device (Local) administrator for the computer account Trusted Root Certificate store, and
- An MDM Agent for any Trusted Root Certificate store.

Thus, FMT_MOF_EXT.1.1 (device user) and FMT_MOF_EXT.1.2 ((device) local administrator or the MDM Agent) represent Windows behavior.

2.2.7.2 Guidance Assurance Activities

None defined.

2.2.7.3 Test Activities

None defined.

2.2.8 Extended: Inter-TSF user data transfer protection (FDP_UPC_EXT.1)

2.2.8.1 TSS Assurance Activity

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions (protection channel) described in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component.

[ST] section 6.3.5 SFR Mapping indicates Windows provides network transport for TLS, HTTPS, Bluetooth BR/EDR, and Bluetooth LE. [ST] identifies the relevant APIs including HttpClient (Appendix B) and Windows.Devices.Bluetooth (Table 18 Device Capabilities).

[ST] section 6.8 Trusted Path Channel identifies interface HttpClient for TLS/HTTPS. For setting/parameter support, see the following from the HttpClient documentation page.

- HttpClient(IHttpFilter), IHttpFilter, Windows.Web.Http.Filters, and HttpBaseProtocolFilter (client certificate, ignorable server certificate errors, server credential)
- SendRequestAsync, HttpRequestMessage, TransportInformation, (server certificate, server certificate errors, server certificate error severity, and server intermediate certificates) RequestUri and Uri
- GetAsync (error E_ILLEGAL_METHOD_CALL) and Uri

[ST] section 6.3.1 Restricting Access to System Services identifies three Bluetooth APIs: Windows.Devices.Bluetooth.GenericAttributeProfile (for LE), Windows.Devices.Bluetooth.Rfcomm (for BR/EDR), and Windows.Devices.Bluetooth (for paired Bluetooth devices). For setting/parameter support, see:

- Windows.Devices.Bluetooth.GenericAttributeProfile, GattCharacteristic, ProtectionLevel (GattProtectionLevel), and CharacteristicProperties (GattCharacteristicProperties)
- Windows.Devices.Bluetooth.Rfcomm, RfcommDeviceService, ProtectionLevel, and SocketProtectionLevel as well as Supporting Bluetooth Devices (XAML) and RFCOMM Scenario: Send File as a Client(XAML)

The evaluator shall examine the TSS to determine that it describes that all protocols listed in the TSS

are specified and included in the requirements in the ST.

[ST] section 5.1.2.6 Extended: Inter-TSF User Data Transfer Protection specifies the protocols: TLS, HTTPS, Bluetooth BR/EDR, and Bluetooth LE. Section 6.3.1 Restricting Access to System Services describes the protocols Bluetooth BR/EDR and LE. Section 6.2.7.1.1 TLS and EAP TLS describes the TLS and HTTPS protocols. Also see guidance section 2.2.8.2 below for description.

2.2.8.2 Guidance Assurance Activities

The evaluator shall confirm that the operational guidance contains instructions necessary for configuring the protocol(s) selected for use by the applications.

[ST] provides references to online documentation for HTTPS/TLS, Bluetooth BR/EDR, and Bluetooth LE, which [Mobile Guide] supplements. [ST] section 6.8 Trusted Path / Channels references HttpClient documentation. HttpClient provides both http and https schemes to applications. [Mobile Guide] section 5 Managing TLS identifies an MDM solution as an option for configuring cipher suites. The section provides links for Windows 10 on how to configure the cipher suites for TLS and EAP-TLS.

[ST] Table 18 Device Capabilities in section 6.3.1 Restricting Access to System Services references online documentation for Windows.Devices.Bluetooth.GenericAttributeProfile and Windows.Devices-.Bluetooth.Rfcomm. Windows.Devices.Bluetooth.GenericAttributeProfile namespace APIs provide applications with access to Bluetooth LE devices. Windows.Devices.Bluetooth.Rfcomm namespace APIs provide support BR/EDR. [Mobile Guide] section 10 Managing Bluetooth states that Bluetooth pairing uses a protected communication channel by default so there is no configuration necessary.

2.2.8.3 Test Activities

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security functions (protection channel) described in these requirements, and verify that the APIs implemented to support this requirement include the appropriate settings/parameters so that the application can both provide and obtain the information needed to assure mutual identification of the endpoints of the communication as required by this component. The evaluator shall write, or the developer shall provide access to, an application that requests protected channel services by the TSF. The evaluator shall verify that the results from the protected channel match the expected results according to the API documentation. This application may be used to assist in verifying the protected channel assurance activities for the protocol requirements.
Test 1: The evaluators shall ensure that the application is able to initiate communications with an external IT entity using each protocol specified in the requirement, setting up the connections as described in the operational guidance and ensuring that communication is successful.
Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext.

These activities were performed in conjunction with FCS_TLS_EXT.1, FCS_TLS_EXT.2, FTP_ITC_EXT.1 and FDP_IFC_EXT.1.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication failure handling (FIA_AFL_EXT.1)

2.3.1.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes that a value corresponding to the number of unsuccessful authentication attempts since the last successful authentication is kept for each user for each Password Authentication Factor interface.

[ST] section 6.4 Identification and Authentication describes how the Local Security Authority component within Windows maintains a count of the consecutive failed logon attempts by security principals from their last successful authentication (search "count of the consecutive failed logon attempts").

The evaluator shall ensure that this description also includes if and how this value is maintained when the TOE is powered off. The evaluator shall ensure that if the value is not maintained, the interface is after another interface in the boot sequence for which the value is maintained.

[ST] section 6.4 Identification and Authentication states that Windows persists the number of consecutive failed logons on for the user and so rebooting the computer does not reset the failed logon counter.

2.3.1.2 Guidance Assurance Activities

The evaluator shall verify that the AGD guidance describes how the administrator configures the maximum number of unsuccessful authentication attempts.

An administrator configures the maximum number of unsuccessful authentication attempts either through an MDM[8] solution or as a local Windows 10 administrator. Subsection 9.1 IT Administrator Guidance describes the MDM policy. Subsection 9.2.1 Local Administrator Guidance provides links to online documentation for a Windows 10 administrator. Subsection 9.3 User Guidance provides a reminder to the user that in the evaluated configuration Windows will wipe a device when the authorization failure limit is exceeded without an option for recovery.

2.3.1.3 Test Activities

The evaluator shall perform the following tests for each available authentication factor interface:

Test 1: *The evaluator shall configure according to the AGD guidance the device with a maximum number of unsuccessful authentication attempts. The evaluator shall enter the locked state and enter incorrect passwords until the wipe occurs. The evaluator shall verify that the number of password*

[8] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

entries corresponds to the configured maximum and that the wipe is implemented.

For Windows 10 Mobile: The evaluator pushed a policy to the TOE that required a wipe after a fixed number of failed authentication attempts. When that number was met, the evaluator observed the TOE rebooted and reinstalled from the factory image, thus wiping the device.

For Windows 10: The evaluator pushed a policy to the TOE that required a wipe after a fixed number of failed authentication attempts. When that number was met, the evaluator observed the TOE rebooted and reinstalled from the factory image, thus wiping the device.

After a failed authentication attempt, Windows displays the warning:

> "If you keep entering the wrong password, you'll be locked out to help protect your data. To unlock, you'll need a BitLocker recovery key."

Please note there is no BitLocker recovery key when Windows 10 is in its evaluated configuration. Consequently, when the maximum number of unsuccessful authentication attempts is reached, recovery is not possible and the device is wiped.

Test 2: The evaluator shall repeat test one, but shall power off (by removing the battery, if possible) the TOE between unsuccessful authentication attempts. The evaluator shall verify that the total number of password entries corresponds to the configured maximum and that the wipe is implemented. Alternatively, if the number of authentication failures is not maintained for the interface under test, the evaluator shall verify that upon booting the TOE between unsuccessful authentication attempts another authentication factor interface is presented before the interface under test.

The evaluator performed this activity similar to Test 1, except reboots the TOE after 2 failed attempts. The evaluator verified that the TOE's authentication failure counter did not reset on reboot and the TOE wiped at the expected threshold.

2.3.2 Bluetooth Authorization and Authentication (FIA_BLT_EXT.1)

2.3.2.1 TSS Assurance Activity

The evaluator shall examine the TSS to ensure that it contains a description of when user permission is required for Bluetooth pairing, and that this description mandates explicit user authorization via manual input for all Bluetooth pairing, including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed.

[ST] section 6.4 Identification and Authentication describes Bluetooth pairing: "The Windows implementation of Bluetooth follows the Bluetooth SIG Specification, including OBEX data transfer RFCOMM, L2CAP, and OPP (object push profile). The OBEX specification, which Windows implements, prevents any transfer of user data until both Bluetooth devices have paired, which requires authorization by the Windows user. When a Windows OS encounters an unpaired device, it does not transfer any data to the unpaired device. When paired to a Bluetooth device, Windows will reject

connection attempts from other devices that purport to use the same Bluetooth address as the connected device."

[ST] section 6.4.3 SFR Mapping states, "Windows requires Bluetooth mutual authentication between the Windows device and the remote device prior to any data transfer over the Bluetooth connection."

The evaluator shall examine the API documentation provided according to Section 6.2.1 and verify that this API documentation does not include any API for programmatic entering of pairing information (e.g. PINs, numeric codes, or "yes/no" responses) intended to bypass manual user input during pairing.

The capability lists in section 6.3.1 and Appendix B the relevant APIs which include enabling and disabling. There are no APIs for programmatic entering of pairing information.

2.3.2.2 Guidance Assurance Activities

The evaluator shall examine the AGD guidance to verify that these user authorization screens are clearly identified and instructions are given for authorizing Bluetooth pairings.

[Mobile Guide] section 10 Managing Bluetooth provides guidance for configuring Bluetooth via MDM or as a Windows 10 local administrator. Subsection 10.1 IT Administrator Guidance provides links example Microsoft configurations. Subsection 10.2.1 Local Administrator Guidance states that Bluetooth is enabled and disabled in the Settings -> Devices -> Bluetooth user interface by setting the radio button labeled Bluetooth to the On or Off state. Sections 10.2.2 User Guidance and 10.3.1 User Guidance provide instructions for Bluetooth pairing for Windows 10 and Windows 10 Mobile, respectively.

If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the operational guidance provides sufficient instruction on limiting the allowed services.

[Mobile Guide] section 10 Managing Bluetooth states "No configuration is necessary to ensure the Bluetooth services provided before login are limited."

2.3.2.3 Test Activities

The evaluator shall perform the following test:

***Test 1**: The evaluator shall perform the following steps:*

***Step 1** - Initiate pairing with the TOE from a remote Bluetooth device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput input-output (IO) capability. (Such a device will attempt to evoke behavior from the TOE that represents the minimal level of user interaction that the TOE supports during pairing.)*

***Step 2** - Verify that the TOE does not permit any Bluetooth pairing without explicit authorization from the user (e.g. the user must have to minimally answer "yes" or "allow" in a prompt).*

The evaluator paired the TOE with a device that requests no man-in-the-middle protection, no bonding, and claims to have NoInputNoOutput capability. The evaluator confirmed that the user must give explicit authorization before pairing.

2.3.3 Bluetooth Authorization and Authentication (FIA_BLT_EXT.1.2)

2.3.3.1 TSS Assurance Activity

None defined.

2.3.3.2 Guidance Assurance Activities

None defined.

2.3.3.3 Test Activities

The evaluator shall perform the following tests for each service protected according to this requirement: ***Test 1:*** *While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a "protected" Bluetooth service (from the second list in the requirement) from a remote device that does not have the required level of trust to use the service. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.*
Test 2: *The evaluator shall repeat Test 1, allow the authorization, and verify that the remote device successfully accesses the service. (Note that this connection may involve pairing, if the untrusted remote device has not yet paired with the TOE.)*

These activities are performed in conjunction with FIA_BLT_EXT.1.1 and FIA_BLT_EXT.2.1.

Test 3: *If the TSF implementation differentiates between trusted and untrusted devices when determining if user authorization is required, repeat Test 1with a service that appears in the second list in the requirement (but not in the first list) and a device that has the required level of trust to use the service. The evaluator shall verify that the user is not prompted for explicit authorization and the connection to the service succeeds.*
Test 4: *If the TSF implementation differentiates between trusted and untrusted devices when determining if user authorization is required, repeat Test 1 with a service that appears in the first list in the requirement and a device that has the required level of trust to use the service. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.*
Test 5: *If the TSF implementation differentiates between trusted and untrusted devices when determining if user authorization is required, repeat Test 2 with a service that appears in the first list*

in the requirement and a device that has the required level of trust to use the service. The evaluator shall verify that the remote device successfully accesses the service if the user explicitly provides authorization.

N/A – These activities are not applicable to the TOE because the TOE does not differentiate between trusted and untrusted devices when determining if user authorization is required.

2.3.4 Extended: Bluetooth Authentication (FIA_BLT_EXT.2)

This requirement was modified by TD0030: Separation of FIA_BLT_EXT.2 Elements. FIA_BLT_EXT.2.2 is now FIA_BLT_EXT.3

2.3.4.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes how data transfer of any type is prevented before the Bluetooth pairing is completed. The TSS shall specifically call out any supported RFCOMM and L2CAP data transfer mechanisms. The evaluator shall ensure that the description in the TSS is detailed enough so that the evaluator can determine that data transfers are only completed after the Bluetooth devices are paired and mutually authenticated.

[ST] section 6.4 Identification and Authentication describes Bluetooth pairing: "The Windows implementation of Bluetooth follows the Bluetooth SIG Specification, including OBEX data transfer, RFCOMM, L2CAP, and OPP (object push profile). The OBEX specification, which Windows implements, prevents any transfer of user data until both Bluetooth devices have paired, which requires authorization by the Windows user. When a Windows OS encounters an unpaired device, it does not transfer any data to the unpaired device."

[ST] section 6.3.1 Restricting Access to System Services includes a description of the Bluetooth RFCOMM capability. The description indicates Windows implements RFCOMM in support of Basic Rate/Extended Data Rate (BR/EDR) transport.

[ST] Section 6.4.3 SFR Mapping states, "Windows requires Bluetooth mutual authentication between the Windows device and the remote device prior to any data transfer over the Bluetooth connection because all Bluetooth profiles are disabled without an explicit authorization by the user."

2.3.4.2 Guidance Assurance Activities

None defined.

2.3.4.3 Test Activities

***Test 1**: The evaluator shall use a Bluetooth tool to attempt to access TOE files using the OBEX Object Push service and verify that pairing and mutual authentication are required by the TOE before allowing access. (If the OBEX Object Push service is unsupported on the TOE, a different service that transfers data over Bluetooth L2CAP and/or RFCOMM may be used in this test.)*

The evaluator attempted to send and receive files to and from the TOE using an external Bluetooth device. The evaluator confirmed that the TOE requires mutual authentication in the form of a PIN before any access is allowed.

2.3.5 Extended: Rejection of Duplicate Bluetooth Connections FIA_BLT_EXT.3

This requirement was modified by TD0030: Separation of FIA_BLT_EXT.2 Elements. FIA_BLT_EXT.3 was FIA_BLT_EXT.2.2

2.3.5.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes how Bluetooth connections are maintained such that two devices with the same Bluetooth device address are not simultaneously connected and such that the initial connection is not superseded by any following connection attempts. The evaluator shall ensure that this description explicitly details the sequence of events that occurs when the TOE receives a new connection request from a device with which it has a current established Bluetooth connection.

When paired to a Bluetooth device, Windows will reject connection attempts from other devices that purport to use the same Bluetooth address as the connected device. (Section 6.4 Identification and Authentication).

2.3.5.2 Guidance Assurance Activities

None defined.

2.3.5.3 Test Activities

The evaluator shall perform the following test:

Test 1*: The evaluator shall perform the following steps:*

Step 1 *- Make a Bluetooth connection between the TOE and a remote Bluetooth device with address a known address (BD_ADDR1).*

Step 2 *- Attempt a connection to the same TOE from a second remote Bluetooth device claiming to have a Bluetooth device address matching BD_ADDR1.*

Step 3 *- Using a Bluetooth protocol analyzer, verify that the second connection attempt is ignored by the TOE and that the initial connection to the device with BR_ADDR1 is unaffected.*

Section 4 and other tables in the PP that list requirement components must be updated to reflect the new component.

The evaluator collected the Bluetooth address of an external device and paired it to the TOE. Using a Bluetooth address spoofing tool, the evaluator attempted to pair a second external device to the TOE using the Bluetooth address of the device that is already paired. The evaluator verified that this attempt failed, and did not affect the initial connection.

2.3.6 Port Access Entity Authentication (FIA_PAE_EXT.1)

2.3.6.1 TSS Assurance Activity

None defined.

2.3.6.2 Guidance Assurance Activities

None defined.

2.3.6.3 Test Activities

The evaluator shall perform the following tests:

Test 1: *The evaluator shall demonstrate that the TOE has no access to the test network. After successfully authenticating with an authentication server through a wireless access system, the evaluator shall demonstrate that the TOE does have access to the test network.*

This activity was performed in conjunction with FCS_TLSC_EXT.1.

Test 2: *The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid client certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.*

The evaluator configured the TOE with an expired client certificate for EAP-TLS. The evaluator attempted an EAP-TLS connection and verified that the TOE rejected the attempt since it did not have a valid certificate.

Test 3: *The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid authentication server certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.*

The evaluator configured the authentication server with an expired EAP-TLS server certificate. The evaluator attempted an EAP-TLS connection and verified that the TOE rejected the attempt since the server did not provide a valid certificate.

2.3.7 Extended: Password Management (FIA_PMG_EXT.1)

2.3.7.1 TSS Assurance Activity

None defined.

2.3.7.2 Guidance Assurance Activities

The evaluator shall examine the operational guidance to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length.

[Mobile Guide] section 11 Managing Passwords provides links to TechNet topics that describe characteristics of strong passwords and best practices for password policy. An administrator configures the minimum password length either through an MDM[9] solution or as a local Windows 10 administrator. Subsection 11.1.1 IT Administrator Guidance describes MDM policy. Subsection 11.1.2 provides a link to online documentation for setting Windows 10 password policy as a local administrator.

2.3.7.3 Test Activities

The evaluator shall also perform the following tests. Note that one or more of these tests can be performed with a single test case.

Test 1: *The evaluator shall compose passwords that either meet the requirements, or fail to meet the requirements, in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.*

The evaluator composed various passwords to meet the requirement. These passwords encompassed the full array of characters that are selectable as well as varying password lengths.

2.3.8 Extended: Authentication Throttling (FIA_TRT_EXT.1)

2.3.8.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes the method by which authentication attempts are not able to be automated.

The evaluator shall ensure that the TSS describes either how the TSF disables authentication via external interfaces (other than the ordinary user interface) or how authentication attempts are delayed in order to slow automated entry and shall ensure that this delay totals at least 500 milliseconds over 10 attempts.

[ST] section 6.4 Identification and Authentication describes how Interactive logons are done on the secure desktop, which does not allow other programs to run, and therefore prevents automated password guessing.

[9] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

[ST] section 6.4 Identification and Authentication describes how the Windows logon component enforces a one second delay between every failed logon with an increased delay after several consecutive logon failures (search “Interactive logons are done on the secure desktop”). In other words, over the course of 10 failed logins, there will be an accumulated delay of at least 10 seconds, which satisfies the minimum delay of 500 milliseconds.

2.3.8.2 Guidance Assurance Activities

None defined.

2.3.8.3 Test Activities

None defined.

2.3.9 Protected Authentication Feedback (FIA_UAU.7)

2.3.9.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes the means of obscuring the password entry.

[ST] section 6.4.3 SFR Mapping describes how during an interactive logon (search “Windows echoes”), Windows echoes the users password with “*” characters to prevent disclosure of the user’s password.

2.3.9.2 Guidance Assurance Activities

The evaluator shall verify that any configuration of this requirement is addressed in the AGD guidance and that the password is obscured by default.

[Mobile Guide] section 11.2 Protecting Passwords states Windows 10 and Windows 10 Mobile do not require any configuration to ensure the password is obscured by default (subsections 11.2.1 Windows 10 and 11.2.2 Windows 10 Mobile, respectively).

2.3.9.3 Test Activities

Test: The evaluator shall enter passwords on the device, including at least the Password Authentication Factor at lockscreen, and verify that the password is not displayed on the device.

From a locked state, the evaluator typed in a password to unlock the TOE and verified it was not displayed. No information regarding the password was displayed to the user.

2.3.10 Extended: Authentication for Cryptographic Operation (FIA_UAU_EXT.1)

2.3.10.1 TSS Assurance Activity

The evaluator shall verify that the TSS section of the ST describes the process for decrypting protected data and keys.

[ST] section 6.4.1 Protecting User Data describes the process for decrypting protected data. Section 6.2.4 Encrypting the Device with BitLocker includes details of BitLocker protection, including process for decrypting protected data and keys. Sections 6.2.5 Key Storage and 6.3.6 Protecting Data with DPAPI cover protecting key data through DPAPI.

The evaluator shall ensure that this process requires the user to enter a Password Authentication Factor and, in accordance with FCS_CKM_EXT.3, derives a KEK which is used to protect the software-based secure key storage and (optionally) DEK(s) for sensitive data, in accordance with FCS_STG_EXT.2.

[ST] section 6.4.1 Protecting User Data states the logon password is used to derive the DPAPI secret (a KEK) which provides an additional layer of protection for certain user data, including keys. Likewise, section 6.4.3 SFR Mapping states for FIA_UAU_EXT.1 that the user must authenticate successfully during interactive logon and prior to decryption of any user data stored on the device

2.3.10.2 Guidance Assurance Activities

None defined.

2.3.10.3 Test Activities

The following tests may be performed in conjunction with FDP_DAR_EXT.1 and FDP_DAR_EXT.2. ***Assurance Activity Note:*** *The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.* ***Test 1:*** *The evaluator shall enable encryption of protected data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as protected data.* *The evaluator shall reboot the device, use a tool provided by developer to search for the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by the developer to access the unique string amongst the application data, and verify that the unique string can be found.*
Test 2: [conditional] *The evaluator shall require user authentication according to the AGD guidance. The evaluator shall store a key in the software-based secure key storage.* *The evaluator shall lock the device, use a tool provided by developer to access the key amongst the stored data, and verify that the key cannot be retrieved or accessed. The evaluator shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the key, and verify that the key can be retrieved or accessed.*
Test 3: [conditional] *The evaluator shall enable encryption of sensitive data and require user authentication according to the AGD guidance. The evaluator shall write, or the developer shall provide access to, an application that includes a unique string treated as sensitive data.* *The evaluator shall lock the device, use a tool provided by developer to attempt to access the unique string amongst the application data, and verify that the unique string cannot be found. The evaluator*

shall enter the Password Authentication Factor to access full device functionality, use a tool provided by developer to access the unique string amongst the application data, and verify that the unique string can be retrieved.

The evaluator enabled device encryption on the TOE and used an application to create a unique string. The evaluator used a tool to view the raw contents of the encrypted drive and verified that a query to find the string did not return a result. The evaluator then authenticated the user and searched for the unique string and succeeded.

2.3.11 Extended: Timing of Authentication (FIA_UAU_EXT.2)

2.3.11.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes the actions allowed by unauthorized users in the locked state.

[ST] section 6.4.3 SFR Mapping describes the only actions that an unauthorized user can take when a Windows device is locked is to bring up the authentication dialog or turn the device off. A Windows 10 Mobile device can place an emergency call or take a photograph.

2.3.11.2 Guidance Assurance Activities

None defined.

2.3.11.3 Test Activities

The evaluator shall verify that the TSS describes the actions allowed by unauthorized users in the locked state. The evaluator shall attempt to perform some actions not listed in the selection while the device is in the locked state and verify that those actions do not succeed.

The evaluator verified that the TOE only allowed actions listed in the ST before authenticating. This test was performed in conjunction with FIA_UAU_EXT.3.

2.3.12 Extended: Re-Authentication (FIA_UAU_EXT.3)

2.3.12.1 TSS Assurance Activity

None defined.

2.3.12.2 Guidance Assurance Activities

None defined.

2.3.12.3 Test Activities

***Test 1**: The evaluator shall configure the TSF to use the Password Authentication Factor according to*

the AGD guidance. The evaluator shall change Password Authentication Factor according to the AGD guidance and verify that the TSF requires the entry of the Password Authentication Factor before allowing the factor to be changed.
***Test 2:** The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF_EXT.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.*
***Test 3:** The evaluator shall configure user-initiated locking according to the AGD guidance. The evaluator shall lock the TSF and then verify that the TSF requires the entry of the Password Authentication Factor before transitioning to the unlocked state.*

The evaluator changed the TOE's password for authentication, locked the TOE (via user-initiated lock) and set an inactivity timeout for the TOE to initiate a lock when it is met. The evaluator observed that the password was required before changing it, and after each instance of the device locking password authentication was required.

2.3.13 Extended: Validation of certificates (FIA_X509_EXT.1)

2.3.13.1 TSS Assurance Activity

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 6.4.2 X.509 Certificate Validation indicates each component that uses X.509 is responsible for certificate validation with a common subcomponent performing the validation. The section describes the certification path validation algorithm by reference to RFC 5280.

See also section 2.2.6 above and [ST] section 6.3.4 VPN Client regarding TOE support of IPsec.

2.3.13.1 Guidance Assurance Activities

None defined.

2.3.13.2 Test Activities

The tests described must be performed in conjunction with the other Certificate Services assurance activities, including the use cases in FIA_X509_EXT.2.1 and FIA_X509_EXT.3. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. 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. ***Test 1:** The evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function (e.g. application validation, trusted channel setup,*

or trusted software update), and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.
Test 2: *The evaluator shall demonstrate that validating an expired certificate results in the function failing.*
Test 3: *The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). For the test of the WLAN use case, only pre-stored CRLs are used. 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.*
Test 4: *The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.*
Test 5: *The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails.*
Test 6: *The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.*
Test 7: *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.)*
Test 8: *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.)*
Test 9: *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 developer provided a custom test app, named x509 Certificate Validation CC Test App, to test these activities. The app runs through a series of tests that test validates a certificate with a good chain, does not validate a certificate with a certificate missing from the chain, does not validate an expired certificate, does not validate a revoked certificate via OCSP and CRL, does not validate a CA certificate that does not contain the basicConstraints extension or does not have the basicConstraints extension set, and validates a CA certificate with the basicConstraints extension to TRUE. The app also modifies the specified bytes in a certificate and the certificate does not validate.

2.3.14 Extended: X509 certificate authentication (FIA_X509_EXT.2)

2.3.14.1 TSS Assurance Activity

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.

[ST] section 6.4.2 describes how each component that uses X.509 certificates will have a repository for public certificates and will select a certificate based on criteria such as entity name for the communication partner, any extended key usage constraints, and cryptographic algorithms associated with the certificate.

[Mobile Guide] Section 13 Managing Certificates provides links to online pages describing configuring the operating environment so that the TOE can use the certificates including importing X.509v3 certificates into the Trust Anchor Database.

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.

[ST] section 6.4.2 X.509 Certificate Validation and Generation covers communication failure during certificate validation for IPsec and TLS (search "if Windows is not able to check the validation status for a certificate"). The TSF actions are:

- Allow the administrator to choose whether to accept the certificate in these cases: HTTPS web browsing
- Allow the user to choose whether to accept the certificate in these cases: HTTPS web browsing
- Not accept the certificate: TLS trusted channel, update Windows, update mobile applications, and integrity verification

For web browsing, a user chooses to accept certificates on a case-by-case basis. The user may be acting as an administrator or standard user.

2.3.14.2 Guidance Assurance Activities

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.

[Mobile Guide] section 13 Managing Certificates covers trusted channel policy. Subsection 13.1 IT Administrator Guidance describes setting Wi-Fi, VPN, and certificate profiles using an MDM as well as providing links to online guidance. Section 13.2 Windows 10 provides the same information for Windows 10 local administrators.

FIA_X509_EXT.2.2 specifies two user behaviors when the TOE cannot establish a connection for revocation checking. The TOE either prevents the connection (EAP-TLS and IPsec) or presents the user with an option to accept the certificate (TLS/HTTPS for web browsing). [Mobile Guide] section 13 covers these behaviors for Windows 10 and Windows 10 Mobile. Subsection 13.1 IT Administrator Guidance describes the first behavior for MDM management. Subsection 13.2.2 Local Administrator Guidance describes local Windows 10 management. Subsection 13.2.3 User Guidance covers the second behavior. In a browsing scenario, Windows presents the user with the option to accept the certificate.

2.3.14.3 Test Activities

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

Test: *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 activity is performed in conjunction with FIA_X509_EXT.1.

2.3.15 Extended: X509 certificate authentication (FIA_X509_EXT.2.3)

2.3.15.1 TSS Assurance Activity

None defined.

2.3.15.2 Guidance Assurance Activities

None defined.

2.3.15.3 Test Activities

None defined.

2.3.16 Extended: X509 certificate authentication (FIA_X509_EXT.2.4)

2.3.16.1 TSS Assurance Activity

If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

When Windows needs to generate a certificate enrollment request it will include a distinguished name, information about the cryptographic algorithms used for the request, any certification extensions, and information about the client requesting the certificate. (Section 6.4.2 X.509 Certificate Validation and Generation).

2.3.16.2 Guidance Assurance Activities

The evaluator shall check to ensure that the operational guidance contains instructions on generating a Certificate Request Message. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the certificate request message.

[Mobile Guide] section 13.2.4 Custom Certificate Requests describes how certificate requests with specific fields such as "Common Name", "Organization", "Organizational Unit", and/or "Country" can

be generated by apps using the Certificates.CertificateEnrollmentManager.CreateRequestAsync API. The section provides a link to the documentation for the API. MDM systems perform certificate enrollment as described in subsection 13.1 IT Administrator Guidance.

2.3.16.3 Test Activities

The evaluator shall also perform the following tests:

Test 1: *The evaluator shall use the operational guidance to cause the TOE to generate a certificate request message. The evaluator shall capture the generated message and ensure that it conforms with the format specified. The evaluator shall confirm that the certificate request provides the public key and other required information, including any necessary user-input information.*

The evaluator requested a certificate from the TOE and noted the required information and public key. The evaluator signed the certificate request and imported it onto the TOE. The evaluator verified that the fields and public key in the certificate matched the ones specified in the request.

Test 2: *The evaluator shall demonstrate that validating a certificate response message without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.*

This activity is performed in conjunction with FIA_X509_EXT.1.

2.3.17 Extended: Request Validation of certificates (FIA_X509_EXT.3)

2.3.17.1 TSS Assurance Activity

The evaluator shall verify that the API documentation provided according to Section 6.2.1 includes the security function (certificate validation) described in this requirement. This documentation shall be clear as to which results indicate success and failure.

[ST] section 6.4.2 X.509 Certificate Validation and Generation identifies interfaces Certificate.BuildChainAsync to construct a certificate chain and CertificateChain.Validate to validate a chain. The API checks are:

1. Certificate.BuildChainAsync
    a. API contract: Windows.Foundation.UniversalApiContract
    b. Success/failure results: always succeeds
2. CertificateChain.Validate
    a. API contract: Windows.Foundation.UniversalApiContract
    b. Success/failure results: ChainValidationResult identifies success and reasons for failures

[ST] section 10 Appendix B: Interfaces and Binaries explicitly states the applicability of the interfaces: "This section is a list of Universal Windows Platform (UWP) APIs used during testing of Windows 10."

2.3.17.2 Guidance Assurance Activities

None defined.

2.3.17.3 Test Activities

The evaluator shall write, or the developer shall provide access to, an application that requests certificate validation by the TSF. The evaluator shall verify that the results from the validation match the expected results according to the API documentation. This application may be used to verify that import, removal, modification, and validation are performed correctly according to the tests required by FDP_STG_EXT.1, FDP_ITC_EXT.1, FMT_SMF_EXT.1.1, and FIA_X509_EXT.1.

This activity is performed in conjunction with FIA_X509_EXT.1.

2.4 Security Management (FMT)

2.4.1 Extended: Management of Security Functions Behavior (FMT_MOF_EXT.1.1)

2.4.1.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes those management functions which may only be performed by the user and confirm that the TSS does not include an Administrator API for any of these management functions. This activity will be performed in conjunction with FMT_SMF_EXT.1.

[ST] Table 9 Management Functions identifies the management functions implemented by the TOE. Functions that the TOE does not implement are struck out. Table 20 Mobile Device Management Capabilities in Section 6.5 Security Management identifies FMT_SMF_EXT.1 management functions that can be performed by a device user, device administrator (local administrator), and MDM agent.

In Table 20, a checkmark means that both Windows 10 and Windows 10 Mobile implement the security function for the particular role. The table also indicates where Windows 10 and Windows 10 Mobile have different behavior, namely, functions 9, 14, 20, 22, 24, 25, and 30. Functions 24 and 30 are available on Windows 10 only.

Section 6.5 explains that Windows does not allow a device user to modify a policy or configuration set by an administrator (which applies to functions 8, 11, 12, 20, 26, 28, 30, 35, and 40).

2.4.1.2 Guidance Assurance Activities

None defined.

2.4.1.3 Test Activities

None defined.

2.4.2 Extended: Management of Security Functions Behavior (FMT_MOF_EXT.1.2)

2.4.2.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes those management functions which may be performed by the Administrator, to include how the user is prevented from accessing, performing, or relaxing the function (if applicable), and how applications/APIs are prevented from modifying the Administrator configuration.

[ST] section 6.5 Security Management presents Table 20 Mobile Device Management Capabilities identifying which FMT_SMF_EXT.1 management functions can be performed by an administrator (local administrator or MDM agent) and administrator-restricted for enrolled devices. Section 6.5 explains Windows does not allow a device user to modify a policy or configuration set by an administrator (which applies to functions 8, 11, 12, 20, 26, 28, and 40).

The TSS also describes any functionality that is affected by administrator-configured policy and how. This activity will be performed in conjunction with FMT_SMF_EXT.1.

Section 6.5 explains Windows does not allow a device user to modify a policy or configuration set by an administrator (which applies to functions 8, 11, 12, 20, 26, 28, and 40).

2.4.2.2 Guidance Assurance Activities

None defined.

2.4.2.3 Test Activities

Test 1: *The evaluator shall use the test environment to deploy policies to Mobile Devices.*
Test 2*: The evaluator shall create policies which collectively include all management functions which are controlled by the (enterprise) administrator and cannot be overridden/relaxed by the user as defined in FMT_MOF_EXT.1.1. The evaluator shall apply these policies to devices, attempt to override/relax each setting both as the user (if a setting is available) and as an application (if an API is available), and ensure that the TSF does not permit it. Note that the user may still apply a more restrictive policy than that of the administrator.*
Test 3: *Additional testing of functions provided to the administrator are performed in conjunction with the testing activities for FMT_SMF_EXT.1.1.*

This activity is performed in conjunction with FMT_SMF_EXT.1. The evaluator configured and tested each management function as specified in the ST.

2.4.3 Extended: Specification of Management Functions (FMT_SMF_EXT.1)

2.4.3.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes all management functions, what role(s) can perform each function, and how these functions are (or can be) restricted to the roles identified by FMT_MOF_EXT.1.

This activity is performed in conjunction with FMT_MOF_EXT.1.1. See sections 2.4.1.1 above and 2.4.2.1 above.

The following activities are organized according to the function number in the table. These activities include TSS assurance activities, AGD assurance activities, and test activities.

Test activities specified below shall take place in the test environment described in the Assurance Activity for FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, and FPT_TUD_EXT.1.3. The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

2.4.3.2 Function 1 TSS Assurance Activity

The evaluator shall verify the TSS defines the allowable policy options: the range of values for both password length and lifetime, and a description of complexity to include character set and complexity policies (e.g., configuration and enforcement of number of uppercase, lowercase, and special characters per password).

Function 1: Configure Password Policy

[ST] section 6.4.3 SFR Mapping states Windows devices support logon passwords at least 14 characters in length up to 127 characters.

[ST] section 6.4.3 states logon passwords can be composed from uppercase characters, lowercase characters, digits, and special characters.

[ST] section 6.6 Security Management states that the complexity requirements include English upper and lowercase characters from A- Z, base 10 digits, non-alphabetic characters, from three of these four categories; and the password lifetime can range from 1 to 999 days.

2.4.3.3 Function 1 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 1: Configure Password Policy

[Mobile Guide] section 11 Managing Passwords provides links to TechNet topics that describe characteristics of strong passwords and best practices for password policy. An administrator configures the minimum password length either through an MDM[10] solution or as a local Windows 10 administrator. Subsection 11.1.1 IT Administrator Guidance describes MDM policy. Subsection 11.1.2 provides a link to online documentation for setting Windows 10 password policy as a local administrator.

2.4.3.4 Function 1 Test Activities

Test 1: *The evaluator shall exercise the TSF configuration as the administrator and perform positive and negative tests, with at least two values set for each variable setting, for each of the following:*

- *minimum password length*
- *minimum password complexity*
- *maximum password lifetime*

Function 1: Configure Password Policy

The evaluator configured the TOE to accept a specified password length and complexity. The evaluator tested a combination of passwords that either met or failed to meet the setting. The evaluator confirmed that the TOE only accepted the passwords that met the setting. The evaluator also configured the password lifetime and observed that when the lifetime was met, the user was forced to change the password.

2.4.3.5 Function 2 TSS Assurance Activity

The evaluator shall verify the TSS defines the range of values for both timeout period and number of authentication failures.

Function 2: Configure Session Locking Policy

[ST] section 6.7 TOE Access describes the session timeout function.

[ST] section 6.4 Identification and Authentication identifies the range of values for the number of consecutive failed logon attempts as from 0 (never lockout the account) to 999.

The timeout can range from 1 minute to 9999 minutes with a default value of 15 minutes.

2.4.3.6 Function 2 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall

[10] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 2: Configure Session Locking Policy

[Mobile Guide] section 16 Locking a Device includes instructions for configuring session locking policy. For Windows 10, section 16.1.1 Local Administrator Guidance covers setting policies for all users. Section 16.1.2 User Guidance covers configuring screen lock timeout. Similarly, section 16.2.1 User Guidance covers screen lock timeout for a Windows 10 Mobile user.

2.4.3.7 Function 2 Test Activities.

Test 2: The evaluator shall exercise the TSF configuration as the user and the administrator. The evaluator shall perform positive and negative tests, with at least two values set for each variable setting, for each of the following.

- *screen-lock enabled/disabled*
- *screen lock timeout*
- *number of authentication failures (may be combined with test for FIA_AFL.1)*

Function 2: Configure Session Locking Policy

This activity was performed in conjunction with FIA_UAU_EXT.3.

2.4.3.8 Function 3 TSS Assurance Activity

None defined.

Function 3: Enable/Disable the VPN Protection

2.4.3.9 Function 3 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 3: Enable/Disable the VPN Protection

[Mobile Guide] section 8 Managing VPN covers configuring VPN, both by an MDM[11] system (section 8.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 8.2.1 Local

[11] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

Administrator Guidance). Evaluation of VPN support is as described in section 2.2.6 Extended: Subset information flow control (FDP_IFC_EXT.1) and [ST] section 6.3.4 VPN Client.

2.4.3.10 Function 3 Test Activities

Test 3: *The evaluator shall perform the following tests:*

Test 3a: *The evaluator shall exercise the TSF configuration to enable the VPN protection. These configuration actions must be used for the testing of the FDP_IFC.1.1 requirement.*

Test 3b: [conditional] *If "per-app basis" is selected, the evaluator shall create two applications and enable one to use the VPN and the other to not use the VPN. The evaluator shall exercise each application (attempting to access network resources; for example by browsing different websites) individually while capturing packets from the TOE. The evaluator shall verify from the packet capture that the traffic from the VPN-enabled application is encapsulated in IPsec and that the traffic from the VPN-disabled application is not encapsulated in IPsec.*

Function 3: Enable/Disable the VPN Protection

This activity was performed in conjunction with FDP_IFC_EXT.1.

2.4.3.11 Function 4 TSS Assurance Activity

The evaluator shall verify that the TSS includes a description of each radio and an indication of if the radio can be enabled/disabled along with what role can do so.

Function 4: Enable/Disable [GPS, Wi-Fi, Bluetooth, mobile broadband]

[ST] section 1.1 Security Target, TOE, and Common Criteria (CC) Identification describes the radios as follows: the Wi-Fi radio is conformant to IEEE 802.11, 3G/4G Mobile Broadband (GSM, WCDMA, and LTE protocol support), and Bluetooth 4.1.

[ST] Table 20 Mobile Device Management Capabilities in Section 6.5 Security Management indicates that the radios can be enabled/disabled by device administrator and MDM agent. Lumia devices used in this evaluation have a GPS radio which provides location services. Enabling/disabling the broadband connection can only be done by the local user and not the mobile device manager. Surface Pro 4 does not have a GPS radio.

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FMT_SMF_EXT.1 Function 4, a footnote indicates only the Microsoft Lumia devices have GPS radios.

In addition the evaluator shall verify that the frequency ranges at which each radio operates is included in the TSS.

[ST] section 6.6.1.1.2 Frequency Ranges provides the required frequency information.

2.4.3.12 Function 4 Guidance Assurance Activities

The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function.

Function 4: Enable/Disable [GPS, Wi-Fi, Bluetooth, mobile broadband]

[Mobile Guide] section 10 Managing Bluetooth provides instructions for configuring Bluetooth, both by an MDM system (section 10.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 10.2.1 Local Administrator Guidance)

[Mobile Guide] section 22 Managing Location Services (GPS) covers enabling/disabling GPS, both by an MDM[12] system (section 22.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 22.2.1 Local Administrator Guidance).

[Mobile Guide] section 23 Managing Wi-Fi covers enabling/disabling Wi-Fi both by an MDM system (section 23.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 23.2.1 Local Administrator Guidance)..

[Mobile Guide] section 28 Managing Mobile Broadband provides a link to user guidance for enabling/disabling mobile broadband (section 28.1 User Guidance).

2.4.3.13 Function 4 Test Activities

Test 4: *The evaluator shall exercise the TSF configuration as both the user and administrator to enable and disable the state of each radio (e.g. Wi-Fi, GPS, cellular, NFC, Bluetooth) listed by the ST author. Additionally, the evaluator shall repeat the steps below, booting into any auxiliary boot mode supported by the device. For each radio, the evaluator shall:*

Step 1 *- Configure spectrum analyzer to sweep desired frequency range for the radio to be tested (based on range provided in the TSS) and place the handset into a Ramsey Box (or other RF-shielding environment) to isolate them from all other RF traffic.*

Step 2 *- The evaluator shall create a baseline of the expected behaviour of RF signals. If a spike of RF activity for the uplink channel for the specific radio frequency band is observed it is deemed that the radio are enabled. The evaluator shall power on the device, ensure the radio in question is enabled, power off the device, enable "Max Hold" on the spectrum analyzer and power on the device. The evaluator shall observe if any RF spikes are present. The evaluator shall enter any necessary passwords to complete the boot process, waiting 2 minutes and resetting the spectrum analyzer between each step.*

Step 3 *- The evaluator shall disable the radio in question and complete the above tests, five times per radio. The evaluator shall verify the absence of RF activity for the uplink channel during device reboot and casual usage.*

Function 4: Enable/Disable [GPS, Wi-Fi, Bluetooth, mobile broadband]

[12] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

The evaluator performed a spectrum analysis on the TOE with each radio enabled and disabled inside a Faraday Bag. The evaluator set a baseline of the spectrum analysis with nothing in the Faraday Bag and compared the result to the capture with the disabled radios. The evaluator confirmed that there were no unexpected spikes in the spectrum analysis when the radios were disabled.

2.4.3.14 Function 5 TSS Assurance Activity

The evaluator shall verify that the TSS includes a description of each collection device and an indication of if it can be enabled/disabled along with what role can do so.

Function 5: Enable/Disable: camera, microphone

[ST] section 6.3.1 Restricting Access to System Services describes the Microphone and the WebCam capability which provides the camera.

[ST] Table 20 Mobile Device Management Capabilities in Section 6.5 Security Management indicates that the camera and microphone can be enabled/disabled across the device by users and administrators.

2.4.3.15 Function 5 Guidance Assurance Activities

The evaluator shall confirm that the AGD guidance describes how to perform the enable/disable function.

Function 5: Enable/Disable: camera, microphone

[Mobile Guide] section 19 Managing Collection Devices describes how to enable/disable the camera and microphone, both by an MDM[13] system (section 21.1 IT Administrator) and by a local Windows 10 administrator (section 19.2.1 Local Administrator Guidance).

2.4.3.16 Function 5 Test Activities

Test 5: *The evaluator shall perform the following test(s):*

Test 5a*: The evaluator shall exercise the TSF configuration as both the user and administrator to enable and disable the state of each audio or visual collection devices (e.g. camera, microphone) listed by the ST author. For each collection device, the evaluator shall disable the device and then attempt to use its functionality. The evaluator shall reboot the TOE and verify that disabled collection devices may not be used during or early in the boot process. Additionally, the evaluator shall boot the device into each available auxiliary boot mode and verify that the collection device cannot be used.*

Test 5b: [conditional] *If "per-app basis" is selected, the evaluator shall create two applications and enable one to use access the A/V device and the other to not access the A/V device. The evaluator*

[13] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

shall exercise each application attempting to access the A/V device individually. The evaluator shall verify that the enabled application is able to access the A/V device and the disabled application is not able to access the A/V device.

Function 5: Enable/Disable: camera, microphone

The evaluator disabled both the camera and microphone on the TOE. The evaluator then attempted to use the camera and microphone through their default apps and verified that TOE denied access. The evaluator confirmed that the TOE allowed access to the camera and microphone via the app when they were enabled.

2.4.3.17 Function 6 TSS Assurance Activity

None defined.

Function 6: Specify wireless networks (SSIDs) to which the TSF may connect

2.4.3.18 Function 6 Guidance Assurance Activities

The evaluator shall specify the wireless network and wireless network settings according to the AGD guidance both as an administrator and as a user.

Function 6: Specify wireless networks (SSIDs) to which the TSF may connect

[Mobile Guide] section 23 Managing Wi-Fi describes how to manage Wi-Fi using an MDM[14] system (section 23.1 IT Administrator).

2.4.3.19 Function 6 Test Activities

The evaluator shall create a test environment consisting of a wireless access system and an authentication server for the purpose of tests associated with functions 6 and 7.

***Test 6:** The evaluator shall specify the wireless network and wireless network settings according to the AGD guidance both as an administrator and as a user. The evaluator shall specify a value for each management function according to the configuration of the test network. Minimally, the evaluator shall construct 2 SSIDs, one corresponding to a WPA2 Enterprise network using EAP-TLS and one corresponding to a disallowed SSID. The evaluator shall verify that the TSF can establish a connection to the allowed SSID, but not to the disallowed SSID.*

Function 6: Specify wireless networks (SSIDs) to which the TSF may connect

This activity was performed in conjunction with FCS_TLSC_EXT.1.

[14] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

2.4.3.20 Function 7 TSS Assurance Activity

The evaluator shall verify the TSS describes the configuration and enforcement of the various credential options used in validation of the WLAN authentication server.

Function 7: Configure the security policy for each wireless network: (a, b, c, d)

Section 6.3.3 Certificate Storage describes how certificates are stored and configured. The Trust Anchor Database contains a list of trusted root Certificate Authority certificates. Access to a certificate store is managed by the discretionary access control policy in Windows such that only the authorized administrator, i.e., the user or the local administrator, can add or remove entries. Certificates which are used by applications, for example, IPsec and TLS, are also placed in certificate stores for the user. Changes to the trusted root certificates can be made using Certificate Trust Lists.

Section 6.5 notes the configuration data for the Wi-Fi settings can be set by the MDM. The policy is enforced when the computer connects to the Wi-Fi network.

2.4.3.21 Function 7 Guidance Assurance Activities

The evaluator shall review the administrative guidance to determine that it describes how to configure the security type, protocol, and client credentials for each of the credential options described in the TSS.

Function 7: Configure the security policy for each wireless network: (a, b, c, d)

[Mobile Guide] section 4 Managing EAP-TLS describes configuration of security policy for wireless networks, both by an MDM[15] system (section 4.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 4.2.1 Local Administrator Guidance).

2.4.3.22 Function 7 Test Activities

The evaluator shall create a test environment consisting of a wireless access system and an authentication server for the purpose of tests associated with functions 6 and 7.

Test 7: *The evaluator shall specify a wireless network with an incorrect value for WLAN authentication server and verify that the Mobile Device cannot connect to the WLAN. The evaluator shall repeat this test, setting incorrect values for the security type and authentication protocol individually and verify that the Mobile Device cannot connect to the WLAN. The evaluator shall then specify, for each credential option claimed in the ST, correct options and demonstrate that the TOE can successfully establish a connection to the WLAN.*

[15] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

Function 7: Configure the security policy for each wireless network: (a, b, c, d)

The evaluator configured separate wireless network profiles each containing an incorrect authentication server, security type, or authentication protocol. The evaluator attempted to connect to each of these profiles and verified that the connection did not succeed. Correct values were tested in conjunction with FCS_TLSC_EXT.1.

2.4.3.23 Function 8 TSS Assurance Activity

None defined.

Function 8: Transition to the locked state

2.4.3.24 Function 8 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 8: Transition to the locked state

[Mobile Guide] section 16 Locking a Device contains instructions for device locking, both by a Windows 10 user (section 16.1.2 User Guidance) and by a Windows 10 Mobile user (section 16.2.1 User Guidance).

2.4.3.25 Function 8 Test Activities

Test 8: The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to transition to a locked state, and verify that the device transitions to the locked state upon command.

Function 8: Transition to the locked state

This activity was tested in conjunction FIA_UAU_EXT.3.

2.4.3.26 Function 9 TSS Assurance Activity

None defined.

Function 9: TSF wipe of protected data

2.4.3.27 Function 9 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 9: TSF wipe of protected data

[ST] Table 20 indicates both Windows 10 and Windows 10 Mobile provide Function 9, but via distinct methods.

[Mobile Guide] section 3 Managing Wipe describes wiping device protected data, both by an MDM system (section 3.1 IT Administrator) and by a local Windows 10 administrator (section 3.2.1 Local Administrator Guidance). In addition, section 3.1 includes guidance for using an MDM system to configure Windows 10 Mobile to wipe a device after a user exceeds a maximum number of consecutive authentication failures.

2.4.3.28 Function 9 Test Activities

Test 9: The evaluator shall use the test environment to instruct the TSF, both as a user and as the administrator, to command the device to perform a wipe of protected data. The evaluator must ensure that this management setup is used when conducting the assurance activities in FCS_CKM_EXT.5.

Function 9: TSF wipe of protected data

This activity was performed in conjunction with FCS_CKM_EXT.5.

2.4.3.29 Function 10 TSS Assurance Activity

The evaluator shall verify the TSS describes the allowable application installation policy options based on the selection included in the ST.

Function 10: Configure application installation policy by (a, c)

[ST] section 6.5 describes restrictions on application installation. (Search “can restrict which applications are installed”.)

If the application whitelist is selected, the evaluator shall verify that the TSS includes a description of each application characteristic upon which the whitelist may be based.

N/A – Application whitelist is not selected.

2.4.3.30 Function 10 Guidance Assurance Activities

The evaluator shall exercise the TSF configuration as the administrator to restrict particular applications, sources of applications, or application installation according to the AGD guidance.

Function 10: Configure application installation policy by (a, c)

[Mobile Guide] section 6 Managing Apps describes how to configure policy for installing applications, both by an MDM[16] system (section 6.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 6.2.1 Local Administrator Guidance).

2.4.3.31 Function 10 Test Activities

Test 10: *Test 10: The evaluator shall exercise the TSF configuration as the administrator to restrict particular applications, sources of applications, or application installation according to the AGD guidance. The evaluator shall attempt to install unauthorized applications and ensure that this is not possible. The evaluator shall, in conjunction, perform the following specific tests:*

Test 10a: [conditional] The evaluator shall attempt to connect to an unauthorized repository in order to install applications.

Test 10b: [conditional] The evaluator shall attempt to install two applications (one whitelisted, and one not) from a known good repository and verify that the application not on the whitelist is rejected. The evaluator shall also attempt to side-load executables or installation packages via USB connections to determine that the white list is still adhered toThe evaluator shall attempt to install unauthorized applications and ensure that this is not possible. The evaluator shall, in conjunction, perform the following specific tests:

Test 10a: [conditional] *The evaluator shall attempt to connect to an unauthorized repository in order to install applications.*

Test 10b: [conditional] *The evaluator shall attempt to install two applications (one whitelisted, and one not) from a known good repository and verify that the application not on the whitelist is rejected. The evaluator shall also attempt to side-load executables or installation packages via USB connections to determine that the white list is still adhered to*

Function 10: Configure application installation policy by (a, c)

The evaluator configured the TOE to not allow installation of apps from the Windows Store. The evaluator attempted to connect to the Windows Store and was unable to access it. The evaluator also attempted to install and use a blacklisted application and verified the TOE denied this attempt. Windows 10 Mobile reported the failure and provided an error code (for example, 0x80073CF9 which generally means an application is not available). The cases where the evaluator was allowed to install an application is tested in conjunction with FPT_TUD_EXT.2.

Note: The TOE is limited to using only Windows Store apps so only blacklisted windows store apps were tested (for example, side-loaded apps were not tested)

[16] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

2.4.3.32 Function 11 and Function 12 TSS Assurance Activity

The evaluator shall verify that the TSS describes each category of keys/secrets that can be imported into the TSF's secure key storage.

Function 11 and 12: Import keys/secrets into the secure key storage, destroy imported keys/secrets and any other keys/secrets in the secure key storage.

[ST] section 6.2.5 Key Storage describes the categories of keys that can be imported. The administrator can configure Certificate Profiles in a Mobile Device Management (MDM) server for importing keys to the enrolled Windows devices. Applications import keys/secrets into the secure key storage by using the CertificateEnrollmentManager.ImportPfxDataAsync API. In addition, on Windows 10 devices users and local administrators can use the Certificate MMC Snap-in to import keys from Personal Information Exchange (.pfx) files into the secure key storage.

2.4.3.33 Function 11 and Function 12 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 11 and 12: Import keys/secrets into the secure key storage, destroy imported keys/secrets and any other keys/secrets in the secure key storage.

See AAR Section 2.1.22.2 in 2.1.22 Extended: Cryptographic Key Storage (FCS_STG_EXT.1) for the guidance on these functions.

2.4.3.34 Function 11 and Function 12 Test Activities

Test 11: & Test 12: The test of these functions is performed in association with FCS_STG_EXT.1.

Function 11 and 12: Import keys/secrets into the secure key storage, destroy imported keys/secrets and any other keys/secrets in the secure key storage.

This activity is covered in conjunction with FCS_STG_EXT.1.

2.4.3.35 Function 13 TSS Assurance Activity

None defined.

Function 13: Import X.509v3 certificates into the Trust Anchor Database

2.4.3.36 Function 13 Guidance Assurance Activities

The evaluator shall review the AGD guidance to determine that it describes the steps needed to import, modify, or remove certificates in the Trust Anchor database, and that the users that have

authority to import those certificates (e.g., only administrator, or both administrators and users) are identified.

Function 13: Import X.509v3 certificates into the Trust Anchor Database

See AAR Section 2.1.22.2 in 2.1.22 Extended: Cryptographic Key Storage (FCS_STG_EXT.1) for the guidance on importing X.509 certificates.

2.4.3.37 Function 13 Test Activities

Test 13: *The evaluator shall import certificates according to the AGD guidance as the user and/or as the administrator, as determined by the administrative guidance. The evaluator shall verify that no errors occur during import. The evaluator should perform an action requiring use of the X.509v3 certificate to provide assurance that installation was completed properly.*

Function 13: Import X.509v3 certificates into the Trust Anchor Database

The evaluator imported a certificate to the TOE and verified no errors occurred. Successful use of this certificate was tested in conjunction with FIA_X509_EXT.1.

2.4.3.38 Function 14 TSS Assurance Activity

The evaluator shall verify that the TSS describes each additional category of X.509 certificates and their use within the TSF.

Function 14: Remove imported X.509v3 certificates and all X.509v3 certificates in the Trust Anchor Database.

[ST] section 6.4.3 SFR Mapping states Windows uses X.509v3 certificates for EAP-TLS exchanges, TLS, HTTPS, code signing for system software updates, code signing for mobile applications, and code signing for integrity verification.

[ST] Table 20 indicates both Windows 10 and Windows 10 Mobile provide Function 14, but via distinct methods.

2.4.3.39 Function 14 Guidance Assurance Activities

The evaluator shall remove an administrator-imported certificate and any other categories of certificates included in the assignment of function 14 from the Trust Anchor Database according to the AGD guidance as the user and as the administrator.

Function 14: Remove imported X.509v3 certificates and all X.509v3 certificates in the Trust Anchor Database.

See AAR Section 2.1.22.2 in 2.1.22 Extended: Cryptographic Key Storage (FCS_STG_EXT.1) for the guidance on removing X.509 certificates.

2.4.3.40 Function 14 Test Activities

Test 14: *The evaluator shall remove an administrator-imported certificate and any other categories of certificates included in the assignment of function 14 from the Trust Anchor Database according to*

the AGD guidance as the user and as the administrator.

Function 14: Remove imported X.509v3 certificates and all X.509v3 certificates in the Trust Anchor Database.

The evaluator removed a certificate from the Trust Anchor Database and verified that the certificate was successfully removed.

2.4.3.41 Function 15 TSS Assurance Activity

The evaluator shall examine the TSS to ensure that it contains a description of each management function that will be enforced by the enterprise once the device is enrolled.

Function 15 Enroll the TOE in management

Table 20 Mobile Device Management Capabilities in Section 6.5 Security Management describes the management functions enforced by the enterprise once the device is enrolled.

2.4.3.42 Function 15 Guidance Assurance Activities

The evaluator shall examine the AGD guidance to determine that this same information is present.

Function 15 Enroll the TOE in management

[Mobile Guide] section 17 Managing Device Enrollment contains instructions for enrolling a mobile device, both for a Windows 10 device (sections 17.1.1 Local Administrator Guidance and 17.1.2 User Guidance) and for a Windows 10 Mobile device (section 17.2.1 User Guidance).

2.4.3.43 Function 15 Test Activities

Test 15: The evaluator shall verify that user approval is required to enroll the device into management.

Function 15 Enroll the TOE in management

This activity was tested in conjunction with FMT_SMF_EXT.2.

2.4.3.44 Function 16 TSS Assurance Activity

The evaluator shall verify that the TSS includes an indication of what applications (e.g., user-installed applications, Administrator-installed applications, or Enterprise applications) can be removed along with what role can do so.

Function 16: Remove applications

[ST] describes removal of applications. (Search “A user is able to uninstall” and “The MDF PP designates Enterprise Applications”.)

[ST] Table 20 Mobile Device Management Capabilities in Section 6.5 Security Management identifies the roles can remove applications as MDM.

2.4.3.45 Function 16 Guidance Assurance Activities

The evaluator shall examine the AGD guidance to determine that it details, for each type of application that can be removed, the procedures necessary to remove those applications and their associated data. For the purposes of this assurance activity, "associated data" refers to data that are created by the app during its operation that do not exist independent of the app's existence, for instance, configuration data, or e-mail information that's part of an e-mail client. It does not, on the other hand, refer to data such as word processing documents (for a word processing app) or photos (for a photo or camera app).

Function 16: Remove applications

[Mobile Guide] section 6 Managing Apps describes how to remove applications, both by an MDM[17] system (section 6.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 6.2.1 Local Administrator Guidance).

2.4.3.46 Function 16 Test Activities

Test 16: *The evaluator shall attempt to remove applications according to the AGD guidance and verify that the TOE no longer permits users to access those applications or their associated data.*

Function 16: Remove applications

The evaluator installed an app onto the TOE that created application data. The data was located on the TOE and the app was then removed. The evaluator verified (by searching the file system) that the application and the application created data was removed.

2.4.3.47 Function 17 TSS Assurance Activity

None defined.

Function 17: Update system software

2.4.3.48 Function 17 Guidance Assurance Activities

The evaluator shall attempt to update the TSF system software following the procedures in the AGD guidance and verify that updates correctly install and that the version numbers of the system software increase.

Function 17: Update system software

[17] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

[Mobile Guide] section 18 Managing Updates describes how to install updates, both by a local Windows 10 administrator (section 18.1.1 Local Administrator Guidance) and by a Windows 10 Mobile user (section 18.2.1 User Guidance).

2.4.3.49 Function 17 Test Activities

Test 17: The evaluator shall attempt to update the TSF system software following the procedures in the AGD guidance and verify that updates correctly install and that the version numbers of the system software increase.

Function 17: Update system software

This activity was performed in conjunction with FPT_TUD_EXT.2.

2.4.3.50 Function 18 TSS Assurance Activity

None defined.

Function 18: Install applications

2.4.3.51 Function 18 Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 18: Install applications

[Mobile Guide] section 6 Managing Apps describes how to install applications, both by an MDM[18] system (section 6.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 6.2.1 Local Administrator Guidance).

2.4.3.52 Function 18 Test Activities

Test 18: The evaluator shall attempt to install a mobile application following the procedures in the AGD guidance and verify that the mobile application is installed and available on the TOE.

Function 18: Install applications

This activity was performed in conjunction with FPT_TUD_EXT.2.

[18] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

2.4.3.53 Function 19 TSS Assurance Activity

The evaluator shall verify that the TSS includes an indication of what Enterprise applications are removable, what actions initiate this removal, and what role can do so. This activity can be performed in conjunction with the TSS activity defined for Function 16.

Function 19: Remove Enterprise Applications

[ST] Table 20 Mobile Device Management Capabilities in Section 6.5 Security Management identifies the roles can remove Enterprise applications as the Administrator and MDM.

2.4.3.54 Function 19 Guidance Assurance Activities

The evaluator shall review the AGD guidance to determine that it describes the steps needed to remove Enterprise applications from the device.

Function 19: Remove Enterprise Applications

[Mobile Guide] section 6 Managing Apps describes how to remove applications, both by an MDM system (section 6.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 6.2.1 Local Administrator Guidance).

2.4.3.55 Function 19 Test Activities

Test 19: *The evaluator shall attempt to remove any Enterprise applications from the device by following the administrator guidance. The evaluator shall verify that the TOE no longer permits users to access those applications or their associated data.*

Function 19: Remove Enterprise Applications

This activity was performed in conjunction with FMT_SMF_EXT.1 Function 16.

2.4.3.56 Function 20 TSS Assurance Activity

The evaluator shall ensure that the TSS includes a description of the Bluetooth profiles and services supported and the Bluetooth security modes and levels supported by the TOE.

Function 20: Configure the Bluetooth trusted channel: (a, b, d)

[ST] Section 6.5 states that Windows does not place any restrictions for the kinds of supported Bluetooth profiles and provides an implementation of Bluetooth Discoverable mode and Low Energy (LE) mode. Section 6.4.3 SFR Mapping includes a link to documentation listing the Bluetooth profiles that Windows 10 supports. Section 13 Appendix D Windows 10 Mobile Bluetooth Profiles covers Windows 10 Mobile. Section 6.4.3 identifies the supported security mode (mode 2) and level (authorization and authentication).

If function c is selected, the evaluator shall verify that the TSS describes any additional wireless technologies that may be used with Bluetooth, including WiFi with Bluetooth High Speed and NFC as an Out of Band pairing mechanism.

Function c is not selected.

If function f is selected, the evaluator shall verify that all supported Bluetooth services are listed in the TSS as manageable and, if the TOE allows disabling by application rather than by service name, that a list of services for each application is also listed.

Function f is not selected.

If function g is selected, the evaluator shall verify that the TSS describes the method by which the level of security for pairings are managed, including whether the setting is performed for each pairing or is a global setting.

Function g is not selected.

If function h is selected, the evaluator shall verify that the TSS describes when Out of Band pairing methods are allowed and which ones are configurable.

Function h is not selected.

2.4.3.57 Function 20 Guidance Assurance Activities

The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 20: Configure the Bluetooth trusted channel: (a, b, d)

[Mobile Guide] section 10 Managing Bluetooth provides instructions for configuring Bluetooth by an MDM system (section 10.1 IT Administrator Guidance) and by a Windows 10 local administrator or user (section 10.2 Windows 10).

2.4.3.58 Function 20 Test Activities

Test 20: *The evaluator shall use a Bluetooth-specific protocol analyzer to perform the following tests of each sub-function:*

Test 20a: *The evaluator shall disable the Discoverable mode and shall verify that other Bluetooth BR/EDR devices cannot detect the TOE. The evaluator shall use the protocol analyzer to verify that the TOE does not respond to inquiries from other devices searching for Bluetooth devices. The evaluator shall enable Discoverable mode and verify that other devices can detect the TOE and that the TOE sends response packets to inquiries from searching devices.*

Test 20b: *The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name, change the Bluetooth device name, and verify that the Bluetooth traffic from the device lists the new name.*

Test 20c: [conditional] *The evaluator shall disable additional wireless technologies for the TOE and verify that the Bluetooth traffic is not able to be sent over WiFi using Bluetooth High Speed, and that NFC cannot be used for pairing. The evaluator shall enable additional wireless technologies and verify that Bluetooth High Speed uses WiFi or that the device can pair using NFC.*

Test 20d: [conditional] *The evaluator shall enable Advertising for Bluetooth LE, verify that the advertisements are captured by the protocol analyzer, disable Advertising, and verify that no advertisements from the device are captured by the protocol analyzer.*

Test 20e: [conditional] *The evaluator shall enable Connectable mode and verify that other Bluetooth devices may pair with the TOE and (if the devices were bonded) re-connect after pairing and disconnection. For BR/EDR devices: The evaluator shall use the protocol analyzer to verify that the TOE responds to pages from the other devices and permits pairing and re-connection. The evaluator shall disable Connectable mode and verify that the TOE does not respond to pages from remote Bluetooth devices, thereby not permitting pairing or re-connection. For LE: The evaluator shall use the protocol analyzer to verify that the TOE sends connectable advertising events and responds to connection requests. The evaluator shall disable Connectable mode and verify that the TOE stops sending connectable advertising events and stops responding to connection requests from remote Bluetooth devices.*

Test 20f: [conditional] *The evaluator shall allow low security modes/levels on the TOE and shall initiate pairing with the TOE from a remote device that allows only something other than Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR), or Security Mode 1/Level 3 (for LE). (For example, a remote BR/EDR device may claim Input/Output capability "NoInputNoOutput" and state that man-in-the-middle (MiTM) protection is not required. A remote LE device may not support encryption.) The evaluator shall verify that this pairing attempt succeeds due to the TOE falling back to the low security mode/level. The evaluator shall then remove the pairing of the two devices, prohibit the use of low security modes/levels on the TOE, then attempt the connection again. The evaluator shall verify that the pairing attempt fails. With the low security modes/levels disabled, the evaluator shall initiate pairing from the TOE to a remote device that supports Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR) or Security Mode 1/Level 3 (for LE). The evaluator shall verify that this pairing is successful and uses the high security mode/level.*

Test 20g: [conditional] *The evaluator shall attempt to pair using each of the Out of Band pairing methods, verify that the pairing method works, iteratively disable each pairing method, and verify that the pairing method fails.*

Function 20: Configure the Bluetooth trusted channel: (a, b, d)

The evaluator used a peer Bluetooth device to scan for discoverable Bluetooth devices. When discoverable mode was disabled on the TOE, it was not found; when discoverable was enabled on the TOE it was found on the peer device. The evaluator used the peer to collect the Bluetooth device name of the TOE. The evaluator confirmed that when the device name was changed on the TOE, the peer recognized the new name. The evaluator used a special app as a listener to analyze the Bluetooth

advertisement traffic from the TOE. The evaluator enabled the listener and observed the traffic from the TOE with advertising enabled. The evaluator then disabled advertising and verified that the listener did not pick up any traffic from the TOE.

2.4.3.59 Function 21 TSS Assurance Activity

None defined.

Function 21: Enable/disable display notification in the locked state of: (a, b, c, d, e)

2.4.3.60 Function 21 Guidance Assurance Activities

The evaluator shall examine the AGD Guidance to determine that it specifies, for at least each category of information selected for Function 21, how to enable and disable display information for that type of information in the locked state.

Function 21: Enable/disable display notification in the locked state of: (a, b, c, d, e)

[Mobile Guide] section 12 Managing Lock Screen Notifications contains instructions to manage notifications on the lock screen, both by a Windows 10 user (sections 12.1.1 Local Administrator Guidance and 12.1.1.1 User Guidance) and by a Windows 10 Mobile user (section 12.2.1 User Guidance).

2.4.3.61 Function 21 Test Activities

Test 21: *For each category of information listed in the AGD guidance, the evaluator shall verify that when that TSF is configured to limit the information according to the AGD, the information is no longer displayed in the locked state.*

Function 21: Enable/disable display notification in the locked state of: (a, b, c, d, e)

The evaluator enabled the selected notifications for the TOE in the locked state. The evaluator observed that when each of these types of messages is pushed to the TOE, the notification is displayed on the locked screen. The evaluator then disabled the notifications and observed that the notifications are no longer displayed on the lock screen.

It should be noted that the following functions are optional capabilities, if the function is implemented, then the following assurance activities shall be performed. The notation of "[conditional] beside the function number indicates that if the function is not included in the ST, then there is no expectation that the assurance activity be performed.

2.4.3.62 Function 22 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS includes a list of each externally accessible hardware port and an indication of if data transfer over that port can be enabled/disabled.

Function 22: Enable/disable all data signaling over [USB hardware ports]

[ST] Section 6.5 indicates only Surface Pro 4 provides the capability to enable and disable data signaling over its USB hardware port.

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FMT_SMF_EXT.1 Function 22, a footnote indicates only Surface Pro 4 provides optional Function 22.

2.4.3.63 Function 22 [Conditional] Guidance Assurance Activities

AGD guidance will describe how to perform the enable/disable function.

Function 22: Enable/disable all data signaling over [USB hardware ports]

[Mobile Guide] section 26 Managing USB describes how to configure USB hardware ports, both by an MDM[19] system (section 26.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 26.2.1 Local Administrator Guidance).

2.4.3.64 Function 22 [Conditional] Test Activities

Test 22: *The evaluator shall exercise the TSF configuration to enable and disable data transfer capabilities over each externally accessible hardware ports (e.g. USB, SD card, HDMI) listed by the ST author. The evaluator shall use test equipment for the particular interface to ensure that no low-level signalling is occurring on all pins used for data transfer when they are disabled. For each disabled data transfer capability, the evaluator shall repeat this test by rebooting the device into the normal operational mode and verifying that the capability is disabled throughout the boot and early execution stage of the device.*

Function 22: Enable/disable all data signaling over [USB hardware ports]

The evaluator attached a logic analyzer to the USB port of the Surface Pro 4. The evaluator then took logic samples while the USB ports were both enabled and disabled and verified that data was passed in the enabled state and no data was transferred while disabled.

2.4.3.65 Function 23 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS describes how the TSF acts as a server in each of the protocols listed in the ST, and the reason for acting as a server.

Function 23: Enable/disable [assignment: list of protocols where the device acts as a server]

Assurance Activity is not applicable. The functionality is not claimed in the security target.

[19] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

2.4.3.66 Function 23 [Conditional] Guidance Assurance Activities

None defined.

Function 23: Enable/disable [assignment: list of protocols where the device acts as a server]

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.67 Function 23 [Conditional] Test Activities

***Test 23**: The evaluator shall attempt to disable each listed protocol in the assignment, which should include tethering uses. The evaluator shall verify that remote devices can no longer access the TOE or TOE resources using any disabled protocols.*

Function 23: Enable/disable [assignment: list of protocols where the device acts as a server]

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.68 Function 24 [Conditional] TSS Assurance Activity

None defined.

Function 24: enable/disable developer modes

2.4.3.69 Function 24 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 24: enable/disable developer modes

[Mobile Guide] section 24 Managing Developer Mode describes how to configure developer mode, both by an MDM[20] system (section 24.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 24.2.1 Local Administrator Guidance).

[ST] Table 20 indicates only Windows 10 limits Function 24 to administrators and MDM.

2.4.3.70 Function 24 [Conditional] Test Activities

***Test 24:** The evaluator shall exercise the TSF configuration as both the user and administrator to enable and disable any developer mode. The evaluator shall test that developer mode access is not*

[20] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

available when its configuration is disabled. The evaluator shall verify the developer mode remains disabled during device reboot.

Function 24: enable/disable developer modes

The evaluator enabled developer mode on the TOE and verified that the user then had access to developer functions. The evaluator then disabled developer mode and verified the user did not have access to developer functions. The evaluator lastly rebooted the TOE and verified that developer mode remained disabled.

2.4.3.71 Function 25 [Conditional] TSS Assurance Activity

None defined.

Function 25: Enable data-at rest protection

2.4.3.72 Function 25 [Conditional] Guidance Assurance Activities

The evaluator shall exercise the TSF configuration as both the user and administrator to enable system-wide data-at-rest protection according to the AGD guidance.

Function 25: Enable data-at rest protection

[ST] Table 20 indicates both Windows 10 and Windows 10 Mobile provide Function 25, but via distinct methods.

See AAR Section 2.2.5.2 in 2.2.5 Extended: Protected Data Encryption (FDP_DAR_EXT.1) for the administrator guidance to enable system-wide data-at-rest protection.

2.4.3.73 Function 25 [Conditional] Test Activities

Test 25: *The evaluator shall exercise the TSF configuration as both the user and administrator to enable system-wide data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all assurance activities for DAR (see Section 0) are conducted with the device in this configuration.*

Function 25: Enable data-at rest protection

This activity was performed in conjunction with FIA_UAU_EXT.1.

2.4.3.74 Function 26 [Conditional] TSS Assurance Activity

None defined.

Function 26: Enable removable media's data-at-rest protection

2.4.3.75 Function 26 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any

configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 26: Enable removable media's data-at-rest protection

See AAR Section 2.2.5.2 in 2.2.5 Extended: Protected Data Encryption (FDP_DAR_EXT.1) for the administrator guidance to enable system-wide data-at-rest protection.

2.4.3.76 Function 26 [Conditional] Test Activities

Test 26: *The evaluator shall exercise the TSF configuration as both the user and administrator to enable removable media's data-at-rest protection according to the AGD guidance. The evaluator shall ensure that all assurance activities for DAR (see Section 0) are conducted with the device in this configuration.*

Function 26: Enable removable media's data-at-rest protection

The evaluator enabled data-at-rest on removable media from the TOE. The evaluator verified that the device was indeed encrypted by looking at raw drive data.

2.4.3.77 Function 27 [Conditional] TSS Assurance Activity

None defined.

Function 27: Enable/disable bypass of local user authentication

2.4.3.78 Function 27 [Conditional] Guidance Assurance Activities

The evaluator shall examine the AGD guidance to determine that it describes how to enable and disable any "Forgot Password", password hint, or remote authentication (to bypass local authentication mechanisms) capability.

Function 27: Enable/disable bypass of local user authentication

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.79 Function 27 [Conditional] Test Activities

Test 27: *For each mechanism listed in the AGD guidance that provides a "Forgot Password" feature or other means where the local authentication process can be bypassed, the evaluator shall disable the feature and ensure that they are not able to bypass the local authentication process.*

Function 27: Enable/disable bypass of local user authentication

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.80 Function 28 [Conditional] TSS Assurance Activity

None defined.

Function 28: wipe Enterprise data

2.4.3.81 Function 28 [Conditional] Guidance Assurance Activities

None defined.

Function 28: wipe Enterprise data

2.4.3.82 Function 28 [Conditional] Test Activities

Test 28: *The evaluator shall attempt to wipe Enterprise data resident on the device according to the administrator guidance. The evaluator shall verify that the data is no longer accessible by the user.*

Function 28: wipe Enterprise data

This activity was performed in conjunction with FCS_CKM_EXT.5.

2.4.3.83 Function 29 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS describes how approval for an application to perform the selected action (import, removal) with respect to certificates in the Trust Anchor Database is accomplished (e.g., a pop-up, policy setting, etc.).

Function 29: Approve [import, removal] by applications of X.509v3 certificates in the Trust Anchor Database

Assurance Activity is not applicable. The functionality is not claimed in the security target.

The evaluator shall also verify that the API documentation provided according to Section 6.2.1 includes any security functions (import, modification, or destruction of the Trust Anchor Database) allowed by applications.

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.84 Function 29 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 29: Approve [import, removal] by applications of X.509v3 certificates in the Trust Anchor Database

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.85 Function 29 [Conditional] Test Activities

Test 29: *The evaluator shall perform one of the following tests:*

Test 29a: [Conditional] *If applications may import certificates to the Trust Anchor Database, the*

evaluator shall write, or the developer shall provide access to, an application that imports a certificate into the Trust Anchor Database. The evaluator shall verify that the TOE requires approval before allowing the application to import the certificate:

○ The evaluator shall deny the approvals to verify that the application is not able to import the certificate. Failure of import shall be tested by attempting to validate a certificate that chains to the certificate whose import was attempted (as described in the Assurance Activity for FIA_X509_EXT.1).

○ The evaluator shall repeat the test, allowing the approval to verify that the application is able to import the certificate and that validation occurs.

***Test 29b: [Conditional]** If applications may remove certificates in the Trust Anchor Database, the evaluator shall write, or the developer shall provide access to, an application that removes certificates from the Trust Anchor Database. The evaluator shall verify that the TOE requires approval before allowing the application to remove the certificate:*

○ The evaluator shall deny the approvals to verify that the application is not able to remove the certificate. Failure of removal shall be tested by attempting to validate a certificate that chains to the certificate whose removal was attempted (as described in the Assurance Activity for FIA_X509_EXT.1).

The evaluator shall repeat the test, allowing the approval to verify that the application is able to remove/modify the certificate and that validation no longer occurs.

Function 29: Approve [import, removal] by applications of X.509v3 certificates in the Trust Anchor Database

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.86 Function 30 [Conditional] TSS Assurance Activity

None defined.

Function 30: Configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate.

2.4.3.87 Function 30 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 30: Configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate.

[ST] Table 20 indicates only Windows 10 supports Function 30.

See AAR section 2.3.14.2 in 2.3.14 Extended: X509 certificate authentication (FIA_X509_EXT.2) for guidance on trusted channel policy. The policy is configurable only for Windows 10.

2.4.3.88 Function 30 [Conditional] Test Activities

Test 30: *The test of this function is performed in conjunction with FIA_X509_EXT.2.2.*

Function 30: Configure whether to establish a trusted channel or disallow establishment if the TSF cannot establish a connection to determine the validity of a certificate.

This activity was performed in conjunction with FIA_X509_EXT.2.

2.4.3.89 Function 31 [Conditional] TSS Assurance Activity

The evaluator shall ensure that the TSS describes which cellular protocols can be disabled.

Function 31: enable/disable the cellular protocols used to connect to cellular network base stations

The LTE broadband protocol in the Lumia devices can be disabled. The Surface Pro 4 does not include a broadband modem.

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FMT_SMF_EXT.1 Function 31, a footnote indicates only the Microsoft Lumia devices have broadband modems.

2.4.3.90 Function 31 [Conditional] Guidance Assurance Activities

The evaluator shall confirm that the AGD guidance describes the procedure for disabling each cellular protocol identified in the TSS.

Function 31: enable/disable the cellular protocols used to connect to cellular network base stations

[Mobile Guide] section 28 Managing Mobile Broadband provides a link to user guidance for enabling/disabling mobile broadband (section 28.1 User Guidance).

2.4.3.91 Function 31 [Conditional] Test Activities

Test 31: *The evaluator shall attempt to disable each cellular protocol according to the administrator guidance. The evaluator shall attempt to connect the device to a cellular network and, using network analysis tools, verify that the device does not allow negotiation of the disabled protocols.*

Function 31: enable/disable the cellular protocols used to connect to cellular network base stations

The evaluator disabled the cellular function on the TOE. The evaluator then attempted to connect to the cellular network and verified that there was not cellular connectivity.

2.4.3.92 Function 32 [Conditional] TSS Assurance Activity

None defined.

Function 32: Read audit logs kept by the TSF

2.4.3.93 Function 32 [Conditional] Guidance Assurance Activities

The evaluator shall attempt to read any device audit logs according to the administrator guidance and verify that the logs may be read.

Function 32: Read audit logs kept by the TSF

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.94 Function 32 [Conditional] Test Activities

Test 32: *The evaluator shall attempt to read any device audit logs according to the administrator guidance and verify that the logs may be read. This test may be performed in conjunction with the assurance activity of FAU_GEN.1.*

Function 32: Read audit logs kept by the TSF

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.95 Function 33 [Conditional] TSS Assurance Activity

None defined.

Function 33: Configure [certificate] used to validate digital signature on applications

2.4.3.96 Function 33 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 33: Configure [certificate] used to validate digital signature on applications

[ST] section 6.6.6.1.1 Windows Store Applications explains Windows Store Applications and their installation packages are verified using a digital signature from Microsoft Corporation with the Code Signing usage. [Mobile Guide] section 13 Managing Certificates covers certificate, import, requests, and enrollment. Subsection 13.1 IT Administrator Guidance describes adding and removing root certificates using an MDM as well as providing links to online guidance. Subsection 13.2 Windows 10 provides the same information for Windows 10 users and local administrators along with instructions for certificate requests. Subsection 13.2.1 Developer Guidance covers how developers implement key management in applications, which applies when users install applications.

2.4.3.97 Function 33 [Conditional] Test Activities

Test 33: *The test of this function is performed in conjunction with FPT_TUD_EXT.2.5.*

Function 33: Configure [certificate] used to validate digital signature on applications

This activity was performed in conjunction with FPT_TUD_EXT.1.

2.4.3.98 Function 34 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS describes how the approval for exceptions for shared use of keys/secrets by multiple applications is accomplished (e.g., a pop-up, policy setting, etc.).

Function 34: Approve exceptions for shared use of keys/secrets by multiple applications

Users and local administrators authorize applications at installation to access shared keys or secrets when an application declares the sharedUserCertificates capability to share the certificate with other Windows Store Applications for the user. The sharedUserCertificates capability is described in [ST] section 6.3.1 Restricting Access to System Services.

2.4.3.99 Function 34 [Conditional] Guidance Assurance Activities

None defined.

Function 34: Approve exceptions for shared use of keys/secrets by multiple applications

2.4.3.100 Function 34 [Conditional] Test Activities

Test 34: *The test of this function is performed in conjunction with FCS_STG_EXT.1.*

Function 34: Approve exceptions for shared use of keys/secrets by multiple applications

2.4.3.101 Function 35 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS describes how the approval for exceptions for destruction of keys/secrets by applications that did not import the key/secret is accomplished (e.g., a pop-up, policy setting, etc.).

Function 35: Approve exceptions for destruction of keys/secrets by applications that did not import the key/secret

Users and local administrators authorize applications at installation to access shared keys or secrets when an application declares the sharedUserCertificates capability to share the certificate with other Windows Store Applications for the user. [ST] section 6.2.5 Key storage explains “Destruction of keys/secrets imported into the secure key storage by applications is conducted automatically by the modern application environment after the keys/secrets are no longer in use.”

2.4.3.102 Function 35 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 35: Approve exceptions for destruction of keys/secrets by applications that did not import the key/secret

[Mobile Guide] section 13.3 Shared User Keys points common application developers to online documentation for Special capabilities, which include the sharedUserCertificates capability.

2.4.3.103 Function 35 [Conditional] Test Activities

Test 35: *The test of this function is performed in conjunction with FCS_STG_EXT.1.*

Function 35: Approve exceptions for destruction of keys/secrets by applications that did not import the key/secret

2.4.3.104 Function 36 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS describes any restrictions in banner settings (e.g., character limitations).

Function 36: Configure the unlock banner

[ST] Section 6.5 states that the banner can use any text string. See footnote for Function 36 in Table 20 Mobile Device Management Capabilities.

2.4.3.105 Function 36 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 36: Configure the unlock banner

[Mobile Guide] section 27 Managing Notifications Prior to Unlocking a Device contains instructions for configuring the unlock banner, both by a Windows 10 administrator (section 27.1.1 Local Administrator Guidance) and by a Windows 10 Mobile user (section 27.2.1 User Guidance).

2.4.3.106 Function 36 [Conditional] Test Activities

Test 36: *The test of this function is performed in conjunction with FTA_TAB.1.*

Function 36: Configure the unlock banner

This activity is performed in conjunction with FTA_TAB.1.

2.4.3.107 Function 37 [Conditional] TSS Assurance Activity

None defined.

Function 37: Configure the auditable items.

2.4.3.108 Function 37 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 37: Configure the auditable items.

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.109 Function 37 [Conditional] Test Activities

Test 37: *The test of this function is performed in conjunction with FAU_SEL.1.*

Function 37: Configure the auditable items.

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.110 Function 38 [Conditional] TSS Assurance Activity

None defined.

Function 38: Retrieve TSF-software integrity verification values

2.4.3.111 Function 38 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 38: Retrieve TSF-software integrity verification values

See AAR Section 2.5.15.2 for the administrator guidance to retrieve TSF software integrity verification values.

2.4.3.112 Function 38 [Conditional] Test Activities

Test 38: *The test of this function is performed in conjunction with FPT_NOT_EXT.1.2.*

Function 38: Retrieve TSF-software integrity verification values

This activity is performed in conjunction with FPT_NOT_EXT.1.

2.4.3.113 Function 39 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS includes a description of how data transfers can be managed

over USB.

Function 39: Enable/Disable USB Mass Storage, USB Data Transfer

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.114 Function 39 [Conditional] Guidance Assurance Activities

None defined.

Function 39: Enable/Disable USB Mass Storage, USB Data Transfer

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.115 Function 39 [Conditional] Test Activities

Test 39: *The evaluator shall perform the following tests based on the selections in 0.*

Test 39a: [conditional] *The evaluator shall disable USB mass storage mode, attach the device to a computer, and verify that the computer cannot mount the TOE as a drive. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.*

Test 39b: [conditional] *The evaluator shall disable USB data transfer without user authentication, attach the device to a computer, and verify that the TOE requires user authentication before the computer can access TOE data. The evaluator shall reboot the TOE and repeat this test with other supported auxiliary boot modes.*

Test 39c: [conditional] *The evaluator shall disable USB data transfer without connecting system authentication, attach the device to a computer, and verify that the TOE requires connecting system authentication before the computer can access TOE data. The evaluator shall then connect the TOE to another computer and verify that the computer cannot access TOE data. The evaluator shall then connect the TOE to the original computer and verify that the computer can access TOE data.*

Function 39: Enable/Disable USB Mass Storage, USB Data Transfer

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.116 Function 40 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS includes a description of available backup methods that can be enabled/disabled.

Function 40: enable/disable backup to [remote system]

[ST] Section 6.5 states that the user can initiate a backup to a remote system which was specified by a MDM. The user can enable/disable backup to a remote system using the "Sync My Settings" settings page.

2.4.3.117 Function 40 [Conditional] Guidance Assurance Activities

None defined.

Function 40: enable/disable backup to [remote system]

2.4.3.118 Function 40 [Conditional] Test Activities

Test 40: *The evaluator shall disable each supported backup location in turn and verify that the TOE cannot complete a backup. The evaluator shall then enable each supported backup location in turn and verify that the TOE can perform a backup.*

Function 40: enable/disable backup to [remote system]

The evaluator disabled the sync feature on the TOE and verified that the sync could not be performed. Then enabled the sync feature and was able to sync the selected settings.

2.4.3.119 Function 41 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS includes a description of Hotspot functionality and USB tethering to include any authentication for these.

Function 41: enable/disable (a. Hotspot, b. USB tethering)

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.120 Function 41 [Conditional] Guidance Assurance Activities

None defined.

Function 41: enable/disable (a. Hotspot, b. USB tethering)

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.121 Function 41 [Conditional] Test Activities

Test 41: *The evaluator shall perform the following tests based on the selections in 0.*

Test 41a: [conditional] *The evaluator shall enable hotspot functionality with each of the of the support authentication methods. The evaluator shall connect to the hotspot with another device and verify that the hotspot functionality requires the configured authentication method.*

Test 41b: [conditional] *The evaluator shall enable USB tethering functionality with each of the of the support authentication methods. The evaluator shall connect to the TOE over USB with another device and verify that the tethering functionality requires the configured authentication method.*

Function 41: enable/disable (a. Hotspot, b. USB tethering)

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.122 Function 42 [Conditional] TSS Assurance Activity

None defined.

Function 42: Approve Exceptions for Sharing Data

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.123 Function 42 [Conditional] Guidance Assurance Activities

None defined.

Function 42: Approve Exceptions for Sharing Data

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.124 Function 42 [Conditional] Test Activities

Test 42: *The test of this function is performed in conjunction with FDP_ACF_EXT.1.2.*

Function 42: Approve Exceptions for Sharing Data

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.125 Function 43 [Conditional] TSS Assurance Activity

None defined.

Function 43: Place Applications into Application Process Groups

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.126 Function 43 [Conditional] Guidance Assurance Activities

None defined.

Function 43: Place Applications into Application Process Groups

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.127 Function 43 [Conditional] Test Activities

Test 43: *The test of this function is performed in conjunction with FDP_ACF_EXT.1.2.*

Function 43: Place Applications into Application Process Groups

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.128 Function 44 [Conditional] TSS Assurance Activity

None defined.

Function 44: Enable/Disable Location Services Across the Device

2.4.3.129 Function 44 [Conditional] Guidance Assurance Activities

The evaluator shall consult the AGD guidance to perform each of the specified tests, iterating each test as necessary if both the user and administrator may perform the function. The evaluator shall verify that the AGD guidance describes how to perform each management function, including any configuration details. For each specified management function tested, the evaluator shall confirm that the underlying mechanism exhibits the configured setting.

Function 44: Enable/Disable Location Services Across the Device

[Mobile Guide] section 22 Managing Location Services (GPS) covers enabling/disabling GPS, both by an MDM[21] system (section 22.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 22.2.1 Local Administrator Guidance). Surface Pro 4 does not have a GPS radio.

2.4.3.130 Function 44 [Conditional] Test Activities

Test 44: *The evaluator shall perform the following tests.*

Test 44a: *The evaluator shall enable location services device-wide and shall verify that an application (such as a mapping application) is unable to access the TOE's location information.*

Test 44b: [conditional] *If "per-app basis" is selected, the evaluator shall create two applications and enable one to use access the location services and the other to not access the location services. The evaluator shall exercise each application attempting to access location services individually. The evaluator shall verify that the enabled application is able to access the location services and the disabled application is not able to access the location services.*

Function 44: Enable/Disable Location Services Across the Device

The evaluator enabled location services and verified that an application on the TOE was able to access the device's location. The evaluator then disabled location and verified that the same app on the TOE was unable to access the TOE's location.

2.4.3.131 Function 45 [Conditional] TSS Assurance Activity

The evaluator shall verify that the TSS describes all assigned security management functions and their intended behavior.

Function 45: No Additional Management Functions

[21] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.132 Function 45 [Conditional] Guidance Assurance Activities

None defined.

Function 45: No Additional Management Functions

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.3.133 Function 45 [Conditional] Test Activities

Test 45: *The evaluator shall design and perform tests to demonstrate that the function may be configured and that the intended behavior of the function is enacted by the TOE.*

Function 45: No Additional Management Functions

Assurance Activity is not applicable. The functionality is not claimed in the security target.

2.4.4 Extended: Specification of Remediation Actions (FMT_SMF_EXT.2)

2.4.4.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes all available remediation actions, when they are available for use, and any other administrator-configured triggers.

[ST] section 6.5.1 SFR Mapping states, "After unenrollment, Windows will remove enterprise applications, and inform the administrator that the device is no longer enrolled." The footnote to FMT_SMF_EXT.2.1 indicates Windows 10 additionally wipes device of protected data.

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FMT_SMF_EXT.2.1, a footnote identifies a remediation action that Windows 10 Mobile provides in addition to actions provided by Windows 10.

2.4.4.2 Guidance Assurance Activities

The evaluator shall unenroll the device according to AGD guidance and verify that the remediation action configured is performed.

[Mobile Guide] section 17 Managing Device Enrollment provides the guidance to unenroll the device, both for Windows 10 (sections 17.1.1 Local Administrator Guidance and 17.1.2 User Guidance) and for Windows 10 Mobile (section 17.2.1 User Guidance).

2.4.4.3 Test Activities

The evaluator shall use the test environment to iteratively configure the device to perform each remediation action in the selection upon unenrollment. The evaluator shall unenroll the device according to AGD guidance and verify that the remediation action configured is performed.

The evaluator enrolled the TOE and issued a certificate to TOE. The evaluator then unenrolled the TOE and verified that the enrollment data was removed as well as the certificate.

2.5 Protection of the TSF (FPT)

2.5.1 Extended: Anti-Exploitation Services (ASLR) (FPT_AEX_EXT.1)

2.5.1.1 TSS Assurance Activity

The evaluator shall ensure that the TSS section of the ST describes how the 8 bits are generated and provides a justification as to why those bits are unpredictable.

[ST] section 6.5.2 Protection from Implementation Weaknesses provides the justification (search "base address is generated").

2.5.1.2 Guidance Assurance Activities

None defined.

2.5.1.3 Test Activities

Assurance Activity Note: *The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

Test 1: *The evaluator shall select 3 apps included with the TSF. These must include any web browser or mail client included with the TSF. For each of these apps, the evaluator will launch the same app on two separate Mobile Devices of the same type and compare all memory mapping locations. The evaluator must ensure that no memory mappings are placed in the same location on both devices.*

If the rare (at most 1/256) chance occurs that two mappings are the same for a single app and not the same for the other two apps, the evaluator shall repeat the test with that app to verify that in the second test the mappings are different.

The evaluator launched 3 apps on 2 identical TOEs and noted their memory mappings. The evaluator verified that the mappings were different on each instance of the TOE.

2.5.2 Extended: Anti-Exploitation Services (ASLR) (FPT_AEX_EXT.1.3)

2.5.2.1 TSS Assurance Activity

None defined.

2.5.2.2 Guidance Assurance Activities

None defined.

2.5.2.3 Test Activities

None defined.

2.5.3 Extended: Anti-Exploitation Services (ASLR) (FPT_AEX_EXT.1.4)

2.5.3.1 TSS Assurance Activity

The evaluator shall ensure that the TSS section of the ST describes how the 4 bits are generated and provides a justification as to why those bits are unpredictable.

[ST] section 6.6.2 Protection from Implementation Weaknesses provides the justification (search "base address is generated").

2.5.3.2 Guidance Assurance Activities

None defined.

2.5.3.3 Test Activities

Assurance Activity Note: *The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

Test 1: *The evaluator shall reboot the TOE at least five times. For each of these reboots, the evaluator shall examine memory mapping locations of the kernel. The evaluator must ensure that no memory mappings are placed in the same location on both devices.*

The evaluator rebooted the TOE 5 times and noted the kernel memory mappings on each reboot. The evaluator confirmed the mappings differed on each reboot.

2.5.4 Extended: Anti-Exploitation Services (Memory Page Permissions) (FPT_AEX_EXT.2.1)

2.5.4.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes of the memory management unit (MMU), and ensures that this description documents the ability of the MMU to enforce read, write, and execute permissions on all pages of virtual memory.

[ST] section 6.6.2 Protection from Implementation Weaknesses states "Windows runs on processors that provide support for virtual memory and enforce restrictions to read, write, and execute pages of virtual and physical memory." [ST] section 6.6.1.1 Supporting Hardware provides Table 21 Supporting Hardware Specifications with the list of processors for each device. In addition, a link is provided for the

hardware specifications and a section identifier is provided so that an individual can identify where to look for information on the MMU.

2.5.4.2 Guidance Assurance Activities

None defined.

2.5.4.3 Test Activities

None defined.

2.5.5 Extended: Anti-Exploitation Services (Memory Page Permissions) (FPT_AEX_EXT.2.2)

2.5.5.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes how the operating system of the application processor prevents all processes executing in a non-privileged execution domain from achieving write and execute permissions on any page of memory (with only specified exceptions).

[ST] section 6.6.1 Separation and Domain Isolation describes memory page protection using Data Execution Prevention (DEP). Search, “execute instructions.”

The evaluator shall ensure that the TSS describes how such processes are unable to request pages of memory with such permissions, and how they are unable to change permissions to both write and execute on any pages already allocated to them.

[ST] Section 6.6.1 describes memory page protection using Data Execution Prevention (DEP) which marks memory pages in a process as non-executable unless the location explicitly contains executable code. The section also describes process isolation for all user-mode processes through private virtual address spaces (private per process page tables), execution context (registers, program counters), and security context (handle table and token). The data structures defining process address space, execution context and security context are all stored in protected kernel-mode memory.

2.5.5.2 Guidance Assurance Activities

None defined.

2.5.5.3 Test Activities

None defined.

2.5.6 Extended: Anti-Exploitation Services (Overflow Protection) (FPT_AEX_EXT.3)

2.5.6.1 TSS Assurance Activity

The evaluator shall determine that the TSS contains a description of stack-based buffer overflow protections implemented in the TSF software which runs in the non-privileged execution mode of the application processor. The exact implementation of stack-based buffer overflow protection will vary by platform. Example implementations may be activated through compiler options such as "-fstack-protector-all", "-fstack-protector", and "/GS" flags.

[ST] section 6.6.2 Protection from Implementation Weaknesses describes stack-based buffer overflow protection (search "stack buffer overrun protection capability"). The section states: "All Windows binaries and Windows Store Applications implement stack buffer overrun protection."

The evaluator shall ensure that the TSS contains an inventory of TSF binaries and libraries, indicating those that implement stack-based buffer overflow protections as well as those that do not. The TSS must provide a rationale for those binaries and libraries that are not protected in this manner.

[ST] section 6.6.2 states that Windows binaries are compiled with stack overflow protection (the /GS compiler option), which is used for all Windows binaries. Microsoft checks that all Windows Store Applications are compiled with buffer overrun protection before ingesting the Windows Store Application into the Windows Store.

2.5.6.2 Guidance Assurance Activities

None defined.

2.5.6.3 Test Activities

None defined.

2.5.7 Extended: Anti-Exploitation Services (Overflow Protection) (FPT_AEX_EXT.3.2)

2.5.7.1 TSS Assurance Activity

The evaluator shall verify that the TSS enumerates the heap implementations provided to userspace processes. The evaluator shall ensure that the TSS lists all types of heap metadata and identifies how the integrity of each type of metadata is ensured.

[ST] Section 6.6.2 describes the heap implementations provided to userspace processes: default allocator and application-implemented allocator. The heap is managed with a collection of metadata (which is not pre-allocated to a specific address), with integrity protection provided by internal checksums and encoding the metadata. If the heap detects corruption due to a heap overrun (e.g. integrity checks fail),

and heap termination on corruption is enabled for the process, then the process is immediately terminated.

The evaluator shall ensure that the TSS identifies all memory address or offset fields within each type of metadata and identifies how the integrity of these addresses or fields is ensured.

[ST] section 6.6.2 states the collection of metadata is not pre-allocated to a specific address. Windows provides integrity protection by internal checksums and encoding the metadata.

The evaluator shall verify that the TSS identifies the manner in which an error condition is entered when a heap overflow is detected and the resulting actions taken by the TSF.

If the heap detects corruption due to a heap overrun (e.g. integrity checks fail), and heap termination on corruption is enabled for the process, then the process is immediately terminated.

2.5.7.2 Guidance Assurance Activities

None defined.

2.5.7.3 Test Activities

For each heap implementation, the evaluator shall write, or the developer shall provide access to, an application which allocates memory from the heap and then writes arbitrary data significantly beyond the end of the allocated buffer. The evaluator shall attempt to execute this application and verify that the write is not allowed.

The evaluator ran an app provided by the developer that attempts to overwrite the memory allocated from the heap. The evaluator confirmed that this attempt is denied and a buffer overflow does not occur.

2.5.8 Extended: Domain Isolation (FPT_AEX_EXT.4)

2.5.8.1 TSS Assurance Activity

The evaluator shall ensure that the TSS describes the mechanisms that are in place that prevents non-TSF software from modifying the TSF software or TSF data that governs the behavior of the TSF. These mechanisms could range from hardware-based means (e.g. "execution rings" and memory management functionality); to software-based means (e.g. boundary checking of inputs to APIs). The evaluator determines that the described mechanisms appear reasonable to protect the TSF from modification.

[ST] sections 6.6.1 Separation and Domain Isolation and 6.6.7 SFR Mapping describe mechanisms protecting TSF software and data (search "by untrusted subject").

The evaluator shall ensure the TSS describes how the TSF ensures that the address spaces of applications are kept separate from one another.

[ST] section 6.6.7 SFR Mapping describes user-mode programs execute in separate virtual address spaces (search "FPT_AEX_EXT.4" in Section 6.6.7).

The evaluator shall ensure the TSS details the USSD and MMI codes available from the dialer at the locked state or during auxiliary boot modes that may alter the behavior of the TSF. The evaluator shall ensure that this description includes the code, the action performed by the TSF, and a justification that the actions performed do not modify user or TSF data. If no USSD or MMI codes are available, the evaluator shall ensure that the TSS provides a description of the method by which actions prescribed by these codes are prevented.

Windows does not have an "auxiliary boot mode" that is distinctly used by a wired interface, as stated in [ST] section 6.6.1.1.3 Mobile Broadband Isolation. Section 6.6.1.1.3 also covers UUSD and MMI codes. Windows 10 does not include the ability to initiate or receive telephony calls. Windows 10 Mobile does not implement any USSD or MMI codes. Hence, entering a USSD or MMI code while the screen is locked has no effect.

The evaluator shall ensure the TSS documents any TSF data (including software, execution context, configuration information, and audit logs) which may be accessed and modified over a wired interface in auxiliary boot modes. The evaluator shall ensure that the description includes data which is modified in support of update or restore of the device. The evaluator shall ensure that this documentation includes the auxiliary boot modes in which the data may be modified, the methods for entering the auxiliary boot modes, the location of the data, the manner in which data may be modified, the data format and packaging necessary to support modification, and software and/or hardware tools, if any, which are necessary for modifying the data.

[ST] Section 6.6.1 states that Windows does not have an "auxiliary boot mode" that is distinctly used by a wired interface.

The evaluator shall ensure that the TSS provides a description of the means by which unauthorized and undetected modification (that is, excluding cryptographically verified updates per FPT_TUD_EXT.2) of the TSF data over the wired interface in auxiliary boots modes is prevented. (The lack of publically available tools is not sufficient justification. Examples of sufficient justification include auditing of changes, cryptographic verification in the form of a digital signature or hash, disabling the auxiliary boot modes, and access control mechanisms that prevent writing to files or flashing partitions.)

[ST] Section 6.6.1 states that Windows does not have an "auxiliary boot mode" that is distinctly used by a wired interface.

2.5.8.2 Guidance Assurance Activities

None defined.

2.5.8.3 Test Activities

Assurance Activity Note: *The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products. In addition, the vendor provides a list of files (e.g., system files, libraries, configuration files, audit logs) that make up the TSF data. This list could be organized by folders/directories (e.g., /usr/sbin, /etc), as well as individual files that may exist outside of the identified directories.*

Test 1: *The evaluator shall check the "permission settings" for each file in vendor provided list of files that make up the TSF and ensure the settings are appropriate for preventing writing by untrusted applications. The evaluator shall attempt to modify a file of their choosing to ensure the mechanism enforces the permission settings and prevents modification.*

The evaluator examined the permission settings in the c:/windows directory and verified that the permission settings were appropriate. The evaluator attempted to modify a file under this directory and verified that this attempt was denied.

Test 2: *The evaluator shall create and load an app onto the Mobile Device. This app shall attempt to traverse over all file systems and report any locations to which data can be written or overwritten. The evaluator must ensure that none of these locations are part of the OS software, device drivers, system and security configuration files, key material, or another application's image/data.*

The evaluator ran an app to enumerate the folders that the app had access to write to and verified that none of these folders were part of the OS, drivers, system and security configuration, keys, or another application's data.

Test 3*: For each available auxiliary boot mode, the evaluator shall attempt to modify a TSF file of their choosing using the software and/or hardware tools described in the TSS. The evaluator shall verify that the modification fails or that the TSF audits the change as expected according to the description in the TSS.*

In Windows 10 the auxiliary boot modes are not applicable in the evaluated configuration, therefore this test case is not applicable.

2.5.9 Application Processor Mediation (FPT_BBD_EXT.1)

2.5.9.1 TSS Assurance Activity

The evaluator shall ensure that the TSS section of the ST describes at a high level how the processors on the Mobile Device interact, including which bus protocols they use to communicate, any other

devices operating on that bus (peripherals and sensors), and identification of any shared resources.

Microsoft claims FPT_BBD_EXT.1 only for Surface Pro 4, as indicated in the footnote to FPT_BBD_EXT.1 in section 5.1.5.5 Extended: Application processor Mediation (FPT_BBD_EXT.1).

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FPT_BBD_EXT.1, a footnote indicates that the security target only claims the optional requirement for Surface Pro 4.

[ST] Section 6.6.1.1.3 Mobile Broadband Isolation states Surface Pro 4 does not include the ability to initiate or receive telephony calls (that is, cellular support). Section 6.6.7 SFR Mapping describes the separation (separate memory, separate cache, and no access to peripherals or sensors) between the application processor and the baseband processor, which provides Wi-Fi and Bluetooth.

The evaluator shall verify that the design described in the TSS does not permit any BPs from accessing any of the peripherals and sensors or from accessing main memory (volatile and non-volatile) used by the AP. In particular, the evaluator shall ensure that the design prevents modification of executable memory of the AP by the BP.

[ST] Section 6.8.1.1.3 Mobile Broadband Isolation states Surface Pro 4 does not include the ability to initiate or receive telephony calls (that is, cellular support). Section 6.6.7 SFR Mapping describes the separation (separate memory, separate cache, and no access to peripherals or sensors) between the application processor and the baseband processor, which provides Wi-Fi and Bluetooth.

2.5.9.2 Guidance Assurance Activities

None defined.

2.5.9.3 Test Activities

None defined.

2.5.10 Extended: Limitation of Bluetooth Profile Support (FPT_BLT_EXT.1)

2.5.10.1 TSS Assurance Activity

None defined.

2.5.10.2 Guidance Assurance Activities

None defined.

2.5.10.3 Test Activities

The evaluator shall perform the following tests:

Test 1: *While the service is not in active use by an application on the TOE, the evaluator shall attempt*

to discover a service associated with a "protected" Bluetooth profile (as specified by the requirement) on the TOE via a Service Discovery Protocol search. The evaluator shall verify that the service does not appear in the Service Discovery Protocol search results. Next, the evaluator shall attempt to gain remote access to the service from a device that does not currently have a trusted device relationship with the TOE. The evaluator shall verify that this attempt fails due to the unavailability of the service and profile.
***Test 2**: The evaluator shall repeat Test 1 with a device that currently has a trusted device relationship with the TOE and verify that the same behavior is exhibited.*

This activity was performed in conjunction with FIA_BLT_EXT.1.

2.5.11 Extended: Key Storage (FPT_KST_EXT.1)

2.5.11.1 TSS Assurance Activity

The evaluator shall consult the TSS section of the ST in performing the assurance activities for this requirement. *In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data.*

[ST] section 6.2.4 Encrypting the Device with BitLocker describes activities from startup (search "prompt the user for the Enhanced PIN"). Section 6.4.1 Protecting User Data describes use of the Enhanced PIN authorization factor for Windows 10. Section 6.2.5 Key Storage covers password authentication, which provides access to private keys and secrets protected by DPAPI (search "When the device is turned on"). Section 6.2.6 Protecting Data with DPAPI provides details of DPAPI access.

The evaluator shall ensure that the description also covers how the cryptographic functions in the FCS requirements are being used to perform the encryption functions, including how the KEKs, DEKs, and stored keys are unwrapped, saved, and used by the TOE so as to prevent plaintext from being written to non-volatile storage. The evaluator shall ensure that the TSS describes, for each power-down scenario how the TOE ensures that all keys in non-volatile storage are wrapped with a KEK.

[ST] sections 6.2.5 Key Storage and 6.2.7 Networking cover unwrapping of keys. See section 2.1.24.1 above in Extended: Integrity of encrypted key storage (FCS_STG_EXT.3), which summarizes the cryptographic algorithms used in unwrapping keys. ST Sections 6.2.4 Encrypting the Device with BitLocker, 6.2.5 Key Storage, and 6.2.6 Protecting Data with DPAPI describe how keys are saved. Windows does not save plain text keys to non-volatile memory. Windows clears keys as summarized above in section 2.1.9.1 above in Cryptographic Key Destruction (FCS_CKM_EXT.4). Windows uses FVEK continuously. Section 6.2.4 covers FVEK clearing on normal shutdown, on hibernation, and on crash.

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the keys) ensure that no unencrypted key material is present in persistent storage.

[ST] section 6.2.4 Encrypting the Device with BitLocker describes encryption of the device's storage unit. Section 6.2.5 Key Storage covers storage of BitLocker keys, which are stored encrypted outside the NTFS partitions (search "FVEK, VMK, and Intermediate Key are stored on disk"). PBKDF is based on CAVP-validated HMAC function (search "The HMAC function forms the basis for").

The evaluator shall review the TSS to determine that it makes a case that key material is not written unencrypted to the persistent storage.

The TSS sections 6.2.4, 6.2.4, and 6.2.7 make the case that the key material is not written unencrypted to persistent storage.

2.5.11.2 Guidance Assurance Activities

None defined.

2.5.11.3 Test Activities

None defined.

2.5.12 Extended: No Key Transmission (FPT_KST_EXT.2)

2.5.12.1 TSS Assurance Activity

The evaluator shall consult the TSS section of the ST in performing the assurance activities for this requirement. The evaluator shall ensure that the TSS describes the TOE security boundary. The cryptographic module may very well be a particular kernel module, the Operating System, the Application Processor, or up to the entire Mobile Device.

[ST] section 6.2.5 Key Storage defines the boundary of the cryptographic module (search "cryptographic module is the combination of the operating system and the device").

In performing their review, the evaluator shall determine that the TSS contains a description of the activities that happen on power-up and password authentication relating to the decryption of DEKs, stored keys, and data.

See section 2.5.11.1 above in Extended: Key Storage (FPT_KST_EXT.1).

The evaluator shall ensure that the TSS describes how other functions available in the system (e.g., regeneration of the keys) ensure that no unencrypted key material is transmitted outside the security boundary of the TOE. The evaluator shall review the TSS to determine that it makes a case that key material is not transmitted outside the security boundary of the TOE.

[ST] section 6.2.5 Key Storage states "No unencrypted BitLocker key material is transmitted outside the cryptographic module." Section 6.2.5 describes handling of intermediate keys, VMK, and FVEK. Section 6.3.6 Protecting Data with DPAPI describes handling and protection of DPAPI keys. Section 6.6.7 SFR Mapping reiterates that plain text keys are not exported from the cryptographic modules

(search “FPT_KST_EXT.2: Plaintext”). The TPM provides protections that prevent the export of TPM data (section 6.2.3 Trusted Platform Module).

2.5.12.2 Guidance Assurance Activities

None defined.

2.5.12.3 Test Activities

None defined.

2.5.13 Extended: No Plaintext Key Export (FPT_KST_EXT.3)

2.5.13.1 TSS Assurance Activity

The ST author will provide a statement of their policy for handling and protecting keys. The evaluator shall check to ensure the TSS describes a policy in line with not exporting either plaintext DEKs, KEKs, or keys stored in the secure key storage.

[ST] section 6.2.5 Key Storage describes the Key Isolation Service, which covers both protected process for handling keys and NTFS files for protecting stored keys.(See also section 6.2.1 Cryptographic Algorithms and Operations.)

Section 6.6.7 SFR Mapping provides a concise summary of policy for handling and protecting keys is as follows:

- During normal operation, Windows does not store plaintext key material in non-volatile storage (FPT_KST_EXT.1).
- Plaintext keys are not exported from the FIPS-validated cryptographic modules (FPT_KST_EXT.2).
- Users cannot export plain text keys from Windows Store applications (FPT_KST_EXT.3).

The policy is in line with not exporting either plaintext DEKs, KEKs, or keys stored in the secure key storage.

2.5.13.2 Guidance Assurance Activities

None defined.

2.5.13.3 Test Activities

None defined.

2.5.14 Extended: Self-Test Notification (FPT_NOT_EXT.1)

2.5.14.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes critical failures that may occur and the actions to be taken upon these critical failures.

[ST] section 6.6.4 Self-Tests describes the start-up self-tests, which are standard FIPS 140-2 cryptographic module tests. Section 6.6.5 Windows Code Integrity describes software integrity tests and Windows' responses to test failures. Windows will fall into a non-operational state after a failure of the Windows FIPS 140 cryptographic self-tests and integrity failure for Windows system binaries (search "FPT_NOT_EXT.1: Windows will fall" in Section 6.6.7 SFR Mapping). Windows will notify the remote administrator via MDM.

2.5.14.2 Guidance Assurance Activities

None defined.

2.5.14.3 Test Activities

Assurance Activity Note: *The following test require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

Test 1: *The evaluator shall use a tool provided by the developer to modify files and processes in the system that correspond to critical failures specified in the second list. The evaluator shall verify that creating these critical failures causes the device to take the remediation actions specified in the first list.*

The evaluator used a kernel debugger attached to the TOE to modify the integrity checking mechanism on boot. The evaluator verified that this modification cause the TOE to go into an error state, attempt to reboot, and audit the event.

2.5.15 Extended: Self-Test Notification (FPT_NOT_EXT.1.2)

2.5.15.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes which critical memory is measured for these integrity values and how the measurement is performed (including which TOE software performs these generates these values, how that software accesses the critical memory, and which algorithms are used)

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FPT_NOT_EXT.1, footnotes identify two optional requirements elements that only apply only when a

device is enrolled as described in the deployment guidance. (In this evaluation, all devices have TPM 2.0.)

[ST] section 6.6.5 Windows Code Integrity describes software integrity tests, how integrity values are measured for each critical memory. Before Windows will unlock the operating system drive, it will verify the integrity of the early boot components, which include the Boot Loader, OS Loader, and OS Resume binaries using file integrity check algorithm. Section 6.4.3 SFR Mapping indicates that X.509v3 certificates are used to support authentication for code signing for integrity verification. Section 6.6.5 describes that when Secure Boot starts in the preboot environment, it will compare the sealed values from the TPM and if those values do not match the calculated values, Secure Boot will lock the system (which prevents booting) and display a warning on the computer display.

After Secure Boot verifies the integrity of early-running kernel components, including Code Integrity, the Code Integrity capability provides measures code integrity for kernel-mode and user-mode programs. Kernel-mode code signing (KMCS) prevents kernel-mode device drivers, such as the BitLocker Drive Encryption Drivers (fvevol.sys), from loading unless they are published and digitally signed by developers who have been vetted by one of a handful of trusted certificate authorities (CAs). KMCS, using public-key cryptography technologies, requires that kernel-mode code include a digital signature generated by one of the trusted certificate authorities. When a kernel device driver tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, but using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain that verifies the signature must match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Windows boot loader.

2.5.15.2 Guidance Assurance Activities

If the integrity values are provided to the administrator, the evaluator shall verify that the AGD guidance contains instructions for retrieving these values and information for interpreting them. (For example, if multiple measurements are taken, what those measurements are and how changes to those values relate to changes in the device state.)

[Mobile Guide] section 25 Managing Health Attestation provides instructions for generating and retrieving health attestation measurements as well as a link to an application for reviewing measurements. The guidance covers both an MDM[22] system (section 25.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 25.2.1 Local Administrator Guidance).

2.5.15.3 Test Activities

Assurance Activity Note: *The following test may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile*

[22] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

Device products.

The evaluator shall repeat the following test for each measurement:

Test: *The evaluator shall boot the device in an approved state and record the measurement taken (either from the log or by using the administrative guidance to retrieve the value via an MDM Agent). The evaluator shall modify the critical memory or value that is measured. The evaluator shall boot the device and verify that the measurement changed.*

The evaluator enrolled the TOE in MDM and sent the attestation record to the MDM admin. Then the evaluator toggled secure boot and sent the attestation record again, verifying that the value had changed.

2.5.16 Extended: Self-Test Notification (FPT_NOT_EXT.1.3)

2.5.16.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes which key the TSF uses to sign the responses to queries and the certificate used to prove ownership of the key. The evaluator shall perform the following test.

Microsoft uses iteration of requirements to identify significant difference in device capability. Microsoft uses footnotes to identify small differences in device capability or evaluation evidence. For FPT_NOT_EXT.1, footnotes identify two optional requirements elements that only apply only when a device is enrolled as described in the deployment guidance. (In this evaluation, all devices have TPM 2.0.)

[ST] Section 6.6.7 SFR Mapping states that when configured to generate health attestations, Windows will use the Attestation Key (AK) in the TPM. Microsoft issues certificates for TPM Attestation Keys.. Table 15 Types of Keys Used by Windows identifies keys used for health attestations as TPM-based RSA keys.

2.5.16.2 Guidance Assurance Activities

None defined.

2.5.16.3 Test Activities

Test: *The evaluator shall write, or the developer shall provide, a management application that queries either the audit logs or the measurements. The evaluator shall verify that the responses to these queries are signed and verify the signatures against the TOE's certificate.*

This activity is performed in conjunction with FPT_NOT_EXT.1.2.

2.5.17 Reliable Time Stamps (FPT_STM.1)

2.5.17.1 TSS Assurance Activity

The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time.

[ST] section 6.6.3 Time Service lists the Windows capabilities that are included the evaluation that use the centralized (i.e., reliable) time service as:

- Network expirations for authentication and data access
- Session timeout and screen locking
- X.509 certificate generation, revocation, and expiration

The TSS provides a description of how the time is maintained and considered reliable in the context of each of the time related functions. This documentation must identify whether the TSF uses a NTP server or the carrier's network time as the primary time sources.

[ST] section 6.6.3 Time Service states each hardware platform supported by the TOE includes a real-time clock as the primary time source. If Windows connects to a broadband network, it will use the network's time server as a secondary time server in the same manner as a domain or a NTP time source.

The real-time clock is a device that can only be accessed using functions provided by the TSF and serves as the reference clock that maintains the system time. Specifically, the TSF provides functions that allow users, including the TSF itself, to query and set the clock, as well as functions to synchronize clocks within a domain. The ability to query the clock is unrestricted, while the ability to set the clock requires the SeSystemtimePrivilege. This privilege is only granted to authorized administrators to protect the integrity of the time service.

Synchronizing the clocks within a managed Windows deployment is critical for cross-machine communications and correlating activities which occur on multiple computers. Windows capabilities that are included in the evaluation and use the centralized (i.e., reliable) time service are:

- Network expirations for authentication and data access
- Session timeout and screen locking
- X.509 certificate generation, revocation, and expiration

Accuracy (which the NIAP OS PP describes as "reliable and monotonically increasing") is described in "*How the Windows Time Service Works*" and a link has been provided.

2.5.17.2 Guidance Assurance Activities

The evaluator examines the operational guidance to ensure it describes how to set the time.

[Mobile Guide] section 14 Managing Time describes setting time manually in Windows 10 (subsection 14.1.1 Local Administrator Guidance) and Windows 10 Mobile (subsection 14.2.1 User guidance). The section covers setting time automatically with an NTP server for Windows 10 (subsection 14.1.1 Local Administrator Guidance) or a mobile operator via Network Identity and Time Zone for Windows 10 Mobile (subsection 14.2.1 user Guidance). Windows 10 Mobile does not support NTP.

2.5.17.3 Test Activities

Test 1: The evaluator uses the operational guide to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

The evaluator queried the time, set the time and queried the time again on the TOE. The evaluator verified that the time was successfully changed.

2.5.18 Extended: TSF Cryptographic Functionality Testing (FPT_TST_EXT.1)

2.5.18.1 TSS Assurance Activity

The evaluator shall examine the TSS to ensure that it specifies the self-tests that are performed at start-up. This description must include an outline of the test procedures conducted by the TSF (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).

[ST] section 6.6.4 Self-Tests lists the Windows start-up self-tests (search “The kernel-mode startup self-tests are”). The tests are as defined in FIPS 140-2 (known answer tests, sign/verify tests, etc.). The section describes the error state where Windows fails to boot (search “If there is a failure in any startup self-test” and in section 6.6.7 SFR Mapping “FPT_TST_EXT.1: Windows runs a series of self-tests”). An administrator enables automatic execution of the start-up self-tests (search “System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing”). [Mobile Guide] section 1.1 specifies this setting as part of the evaluated configuration.

The TSS must include any error states that they TSF may enter when self-tests fail, and the conditions and actions necessary to exit the error states and resume normal operation.

[ST] section 6.6.4 Self-Tests describes the error state where Windows fails to boot (search “If there is a failure in any startup self-test” and in section 6.6.7 SFR Mapping “FPT_TST_EXT.1: Windows runs a series of self-tests”).

The evaluator shall verify that the TSS indicates these self-tests are run at start-up automatically, and do not involve any inputs from or actions by the user or operator.

[ST] section 6.6.4 Self-Tests describes that an administrator enables automatic execution of the start-up self-tests (search “System Cryptography: Use FIPS compliant algorithms for encryption, hashing, and signing”). [Mobile Guide] section 1.1 specifies this setting as part of the evaluated configuration.

The evaluator shall inspect the list of selftests in the TSS and verify that it includes algorithm self-tests. The algorithm self-tests will typically be conducted using known answer tests.

[ST] section 6.6.4 Self-Tests lists the Windows start-up self-tests (search “The kernel-mode startup self-tests are”). The tests are as defined in FIPS 140-2 cryptographic module algorithm or known answer tests.

2.5.18.2 Guidance Assurance Activities

None defined.

2.5.18.3 Test Activities

None defined.

2.5.19 Extended: TSF Integrity Testing (FPT_TST_EXT.2.1)

2.5.19.1 TSS Assurance Activity

The evaluator shall verify that the TSS section of the ST includes a description of the boot procedures, including a description of the entire bootchain, of the software for the TSF's Application Processor. The evaluator shall ensure that before loading the bootloader(s) for the operating system and the kernel, all bootloaders and the kernel software itself is cryptographically verified. For each additional category of executable code verified before execution, the evaluator shall verify that the description in the TSS describes how that software is cryptographically verified.

[ST] section 6.6.5 Windows Code Integrity describes the Window boot procedures including the entire bootchain, and its integrity mechanisms. Secure Boot capability of Windows verifies the integrity of the early boot components. Secure Boot relies on file measurements sealed to the TPM. Section 6.2.3 Trusted Platform Module describes TPM sealing including hardware protection of the Storage Root Key. Windows verifies the integrity of kernel software through its code integrity capability (search "Code Integrity capability provides measures code integrity for kernel-mode and user-mode programs"). Section 6.2.1 Cryptographic Algorithms and Operations states, "FIPS 140 AES-256 Counter Mode DBRG Known Answer Tests (instantiate, generate) on start-up. Windows always runs the SP 800-90-mandated self-tests for AES-CTR-DRBG during a reseed and runs the Dual-EC reseed self-test when the user chooses to operate Windows in the FIPS validated mode." Section 6.6.1 Separation and Domain Isolation indicates that the TOE's Code Integrity Verification feature use the FIPS-certified cryptographic libraries.

The evaluator shall verify that the TSS contains a justification for the protection of the cryptographic key or hash, preventing it from being modified by unverified or unauthenticated software. The evaluator shall verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

[ST] section 6.2.3 Trusted Platform Module describes protection of the TPM key. Section 6.2.5 Key Storage describes how keys are protected. Section 6.6.5 Windows Code Integrity describes how Kernel-mode code signing (KMCS) prevents kernel-mode device drivers, such as the BitLocker Drive Encryption Drivers (fvevol.sys), from loading unless they are published and digitally signed by developers who have been vetted by one of a handful of trusted certificate authorities (CAs). When a kernel device driver tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, but using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain that verifies the signature must

match one of the Microsoft's root public keys indicating that Microsoft is the publisher of the Windows image files. These Microsoft's root public keys are hardcoded in the Windows boot loader.

The evaluator shall verify that the TSS describes each auxiliary boot mode available on the TOE during the boot procedures. The evaluator shall verify that, for each auxiliary boot mode, a description of the cryptographic integrity of the executed code through the kernel is verified before each execution.

[ST] section 6.6.5 Windows Code Integrity describes the Window boot procedures including the entire bootchain, and its integrity mechanisms. When Secure Boot starts in the preboot environment, it will compare the sealed values from the TPM and if those values do not match the calculated values, Secure Boot will lock the system (which prevents booting) and display a warning on the computer display. Next the Code Integrity capability provides code integrity checking for kernel-mode programs.

2.5.19.2 Guidance Assurance Activities

None defined.

2.5.19.3 Test Activities

The evaluator shall perform the following tests:

Test 1: *The evaluator shall perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the TOE properly boots.*

This activity is performed in conjunction with FPT_NOT_EXT.1.

Assurance Activity Note: *The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

Test 2: *The evaluator shall modify a TSF executable that is integrity protected and cause that executable to be successfully loaded by the TSF. The evaluator observes that an integrity violation is triggered and the TOE does not boot. (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that the module was modified so that it was rendered unable to run because its format was corrupt).*

The evaluator loaded a modified version of a boot file such onto the TOE. The evaluator confirmed that the integrity violation is triggered on boot and the TOE fails to load.

Assurance Activity Note: *The following tests require the vendor to provide access to a test platform that provides the evaluator with tools that are typically not found on consumer Mobile Device products.*

[conditional] Test 3: *If the ST author indicates that the integrity verification is performed using a public key, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1. The evaluator shall digitally sign the TSF executable with a certificate that does not have the Code Signing purpose in the extendedKeyUsage field and verify that an integrity violation is triggered. The evaluator shall repeat the test using a certificate that contains the Code Signing purpose and verify that the integrity verification succeeds. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*

This activity is performed in conjunction with FPT_TUD_EXT.2.

2.5.20 Extended: TSF Integrity Testing (FPT_TST_EXT.2.2)

2.5.20.1 TSS Assurance Activity

None defined.

2.5.20.2 Guidance Assurance Activities

None defined.

2.5.20.3 Test Activities

Testing for this element are performed in conjunction with the assurance activities for FPT_TST_EXT.2.1.

2.5.21 Extended: Trusted Update: TSF Version Query (FPT_TUD_EXT.1)

2.5.21.1 TSS Assurance Activity

None defined.

2.5.21.2 Guidance Assurance Activities

Using the AGD guidance provided, the evaluator shall test that the administrator and user can query:

- *the current version of the TSF operating system and any firmware that can be updated separately*
- *the hardware model of the TSF*
- *the current version of all installed mobile applications*

[Mobile Guide]section 15 Getting Version Information provides instructions to determine the hardware model and operating system version, both for Windows 10 (subsection 15.1.1 User Guidance) and for Windows 10 Mobile (subsection (15.2.1 User Guidance).

2.5.21.3 Test Activities

The evaluator shall establish a test environment consisting of the Mobile Device and any supporting software that demonstrates usage of the management functions. This can be test software from the developer, a reference implementation of management software from the developer, or other commercially available software. The evaluator shall set up the Mobile Device and the other software to exercise the management functions according to provided guidance documentation.

Test 1: *Using the AGD guidance provided, the evaluator shall test that the administrator and user can query:*

- *the current version of the TSF operating system and any firmware that can be updated separately*
- *the hardware model of the TSF*
- *the current version of all installed mobile applications*

The evaluator must review manufacturer documentation to ensure that the hardware model identifier is sufficient to identify the hardware which comprises the device.

The evaluator successfully queries the TOE for the TSF OS, firmware, hardware model, and current version of all applications.

2.5.22 Extended: Trusted Update Verification (FPT_TUD_EXT.2)

2.5.22.1 TSS Assurance Activity

The evaluator shall verify that the TSS section of the ST describes all TSF software update mechanisms for updating the system software. The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails.

[ST] section 6.6.6 Windows and Application Updates describes the software update mechanisms. Software update includes digital signature verification (search "are signed by Microsoft" and "otherwise the installation will abort"). The section states "The integrity of the Microsoft Code Signing certificate on the computer is protected by the storage root key within the TPM". Secure Boot verifies the integrity of the boot loader (see sections 2.5.15 above and 2.5.19 above). Microsoft's root public keys are hardcoded in the Windows boot loader.

The evaluator shall verify that all software and firmware involved in updating the TSF is described and, if multiple stages and software are indicated, that the software/firmware responsible for each stage is indicated and that the stage(s) which perform signature verification of the update are identified.

Integrity checking of both the Microsoft Update Packages and Windows executable code, and the Windows Store Applications and their installation packages are verified using a digital signature from Microsoft Corporation with the Code Signing usage. The process of verification is the same for all update types and is described in Section 6.6.6 Windows and Application Updates.

The evaluator shall verify that the TSS describes the method by which the digital signature is verified and that the public key used to verify the signature is either hardware-protected or is validated to chain to a public key in the Trust Anchor Database. If hardware-protection is selected, the evaluator shall verify that the method of hardware-protection is described and that the ST author has justified why the public key may not be modified by unauthorized parties.

[ST] section 6.6.6 Windows and Application Updates describes the software update mechanisms. Software update includes digital signature verification (search "are signed by Microsoft" and "otherwise the installation will abort"). The section states "The integrity of the Microsoft Code Signing certificate on the computer is protected by the storage root key within the TPM". Secure Boot verifies the integrity of the boot loader (see above sections 2.5.15 above and 2.5.19 above). Microsoft's root public keys are hardcoded in the Windows boot loader.

***[conditional]** If the ST author indicates that software updates to system software running on other processors is verified, the evaluator shall verify that these other processors are listed in the TSS and that the description includes the software update mechanism for these processors, if different than the update mechanism for the software executing on the Application Processor.*

N/A – the ST author did not indicate that software updates to system software running on other processors is verified.

***[conditional]** If the ST author indicates that the public key is used for software update digital signature verification, the evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.*

Section 6.4.3 SFR Mapping indicates that X.509v3 certificates are used to support software update digital signature verification. Section 6.5.2 X.509 Certificate Validation describes certificate validation in accordance with FIA_X509_EXT.1 including all applicable usage constraints (Code Signing purpose) in the extendedKeyUsage as described in RFC 5280.

2.5.22.2 Guidance Assurance Activities

None defined.

2.5.22.3 Test Activities

The evaluator shall verify that the developer has provided evidence that the following tests were performed for each available update mechanism:

***Test 1:** The tester shall try to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.*

The evaluator attempted to install an update without a digital signature and verified that this attempt failed. The evaluator successfully installed an update with a valid digital signature.

Test 2: *The tester shall digitally sign the update with a key disallowed by the device and verify that installation fails. The tester shall digitally sign the update with the allowed key and verify that installation succeeds.*

The evaluator attempted to install an update with and untrusted "disallowed" key and verified the attempt failed. The successful case was tested in Test 1.

Test 3: [conditional] *The tester shall digitally sign the update with an invalid certificate and verify that update installation fails. The tester shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. The tester shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the application installation succeeds.*

The evaluator attempted to install 2 updates; one with an invalid certificate and the other with a valid certificate with no code signing purpose and verified both these attempts failed. The successful cases were tested in Test 1.

Test 4: [conditional] *The tester shall repeat this test for the software executing on each processor listed in the first selection. The tester shall attempt to install an update without the digital signature and shall verify that installation fails. The tester shall attempt to install an update with digital signature, and verify that installation succeeds.*

N/A—no selection made in the Security Target.

2.5.23 Extended: Trusted Update Verification (FPT_TUD_EXT.2.4)

2.5.23.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes how mobile application software is verified at installation. The evaluator shall ensure that this method uses a digital signature.

[ST] section 6.6.6.1.1 Windows Store Applications and their installation packages are verified using a digital signature from Microsoft Corporation with the Code Signing usage.

2.5.23.2 Guidance Assurance Activities

None defined.

2.5.23.3 Test Activities

Test 1: *The evaluator shall write, or the developer shall provide access to, an application. The evaluator shall try to install this application without a digitally signature and shall verify that installation fails. The evaluator shall attempt to install a digitally signed application, and verify that installation succeeds.*

The evaluator attempted to install an application without a digital signature and verified that this attempt failed. The evaluator successfully installed an application with a valid digital signature along with the activity for FPT_TUD_EXT.2.7.

2.5.24 Extended: Trusted Update Verification (FPT_TUD_EXT.2.5)

2.5.24.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes how mobile application software is verified at installation. The evaluator shall ensure that this method uses a digital signature by a code signing certificate.

[ST] section 6.6.6.1.1 Windows Store Applications and their installation packages are verified using a digital signature from Microsoft Corporation with the Code Signing usage.

2.5.24.2 Guidance Assurance Activities

None defined.

2.5.24.3 Test Activities

Test 1: *The evaluator shall write, or the developer shall provide access to, an application. The evaluator shall try to install this application without a digitally signature and shall verify that installation fails. The evaluator shall attempt to install an application digitally signed with an appropriate certificate, and verify that installation succeeds.*

This activity was performed in conjunction with FPT_TUD_EXT.2.4.

Test 2: *The evaluator shall digitally sign the application with an invalid certificate and verify that application installation fails. The evaluator shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. This test may be performed in conjunction with the assurance activities for FIA_X509_EXT.1.*

The evaluator attempted to install 2 applications; one with an invalid certificate and the other with a valid certificate with no code signing purpose and verified both these attempts failed. The successful cases were tested in FPT_TUD_EXT.2.7.

Test 3: *If necessary, the evaluator shall configure the device to limit the public keys that can sign application software according to the AGD guidance. The evaluator shall digitally sign the application with a certificate disallowed by the device or configuration and verify that application installation fails. The evaluator shall attempt to install an application digitally signed with an authorized certificate and verify that application installation succeeds.*

The evaluator attempted to install an application with and untrusted “disallowed” key and verified the attempt failed. The successful case was tested with FPT_TUD_EXT.2.7.

2.5.25 Extended: Trusted Update Verification (FPT_TUD_EXT.2.6)

2.5.25.1 TSS Assurance Activity

None defined.

2.5.25.2 Guidance Assurance Activities

None defined.

2.5.25.3 Test Activities

Testing for this element are performed in conjunction with the assurance activities for FPT_TUD_EXT.2.3 and FPT_TUD_EXT.2.5.

This activity was performed in conjunction with FPT_TUD_EXT.2.3 and FPT_TUD_EXT.2.5.

2.5.26 Extended: Trusted Update Verification (FPT_TUD_EXT.2.7)

2.5.26.1 TSS Assurance Activity

The evaluator shall verify that the TSS describes the mechanism that prevents the TSF from installing software updates that are an older version that the currently installed version.

[ST] section 6.6.6 Windows and Application Updates indicates that the Windows installer will not install an update if the files in the package have lower version numbers than the installed files.

2.5.26.2 Guidance Assurance Activities

None defined.

2.5.26.3 Test Activities

The evaluator shall repeat the following tests to cover all allowed software update mechanisms as described in the TSS. For example, if the update mechanism replaces an entire partition containing many separate code files, the evaluator does not need to repeat the test for each individual file.

***Test 1:** The evaluator shall attempt to install an earlier version of software (as determined by the manufacturer). The evaluator shall verify that this attempt fails by checking the version identifiers or cryptographic hashes of the privileged software against those previously recorded and checking that the values have not changed.*

The evaluator installed an application on the TOE and attempted to install an older version of the application and verified this attempt was denied.

Test 2: *The evaluator shall attempt to install a current or later version and shall verify that the update succeeds.*

The evaluator attempted to install a newer (later) version of the application from Test 1. The evaluator observed that this attempt succeeded.

2.6 TOE Access (FTA)

2.6.1 Extended: TSF- and User-initiated locked state (FTA_SSL_EXT.1)

2.6.1.1 TSS Assurance Activity

The evaluator shall verify the TSS describes the actions performed upon transitioning to the locked state.

[ST] describes the transition to locked state in Section 6.7 TOE Access describes how when the screen is locked, Windows displays a screen saver (search "lock the workstation and execute the screen saver"), and shows notifications from applications which have registered to publish notifications to the locked screen (search "notifications from applications which have registered").

[ST] section 6.7 describes information displayed to users through notifications:

> "… if the workstation was not locked manually, the TSF will lock the display and start the screen saver program if and when the inactivity period is exceeded, as well any notifications from applications which have registered to publish the application's badge or the badge with associated notification text to the locked screen. The user has the option to not display any notifications, or choose one Windows Store Application to display notification text, and select other applications display their badge.
>
> For Windows 10 the inbox Calendar, Weather, and Alarm applications can generate notifications, and when selected to display notification text they will show the location and time of the upcoming and in-progress meeting, the current weather conditions, and an expired alarm times. In addition, Mail application can be configured to display a badge but not notification text.
>
> For Windows 10 Mobile the inbox Calendar, Mail, [SMS] Messaging, and Phone applications can generate notifications, and when selected to display notification text they will show the location and time of the upcoming and in-progress meeting, the sender and subject line of the last received email, the sender and text from the last received SMS message, and the last phone caller and caller notification respectively."

The evaluator shall verify that the TSS describes the information allowed to be displayed to unauthorized users.

Section 6.4.3 SFR Mapping indicates that when the Windows device is locked an unauthorized user only see the authentication (search "FIA_UAU_EXT.2: The only actions").

2.6.1.2 Guidance Assurance Activities

The evaluation shall verify that the AGD guidance describes the method of setting the inactivity

interval and of commanding a lock.

[Mobile Guide] section 16 Locking a Device provides instructions for setting inactivity interval, both for Windows 10 (subsections 16.1.1 Local Administrator Guidance and 16.1.2 User Guidance) and for Windows 10 Mobile (subsection 16.2.1 User Guidance). Section 16 contains instructions for device locking, both by a Windows 10 user (subsection 16.1.2 User Guidance) and by a Windows 10 Mobile user (subsection 16.2.1 User Guidance).

2.6.1.3 Test Activities

Test 1: *The evaluator shall configure the TSF to transition to the locked state after a time of inactivity (FMT_SMF_EXT.1) according to the AGD guidance. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.*
Test 2: *The evaluator shall command the TSF to transition to the locked state according to the AGD guidance as both the user and the administrator. The evaluator shall wait until the TSF locks and verify that the display is cleared or overwritten and that the only actions allowed in the locked state are unlocking the session and those actions specified in FIA_UAU_EXT.2.*

This activity was performed in conjunction with FIA_UAU_EXT.3.

2.6.2 Default TOE Access Banners (FTA_TAB.1)

2.6.2.1 TSS Assurance Activity

The TSS shall describe when the banner is displayed.

[ST] section 2.2 TOE Security Services states that the system can be configured to display a logon banner before the logon dialog. Section 6.7 states "As part of establishing the interactive logon session, Windows can be configured to display a logon banner, which is specified by the administrator, that the user must accept prior to establishing the session."

2.6.2.2 Guidance Assurance Activities

None defined.

2.6.2.3 Test Activities

The evaluator shall also perform the following test: ***Test 1:*** *The evaluator follows the operational guidance to configure a notice and consent warning message. The evaluator shall then start up or unlock the TSF. The evaluator shall verify that the notice and consent warning message is displayed in each instance described in the TSS.*

The evaluator configured the notice and consent waring on the TOE. The evaluator then locked the TOE and verified that the warning was displayed.

2.6.3 Extended: Wireless Network Access (FTA_WSE_EXT.1)

2.6.3.1 TSS Assurance Activity

The assurance activity for this requirement is performed in conjunction with the assurance activity for FMT_SMF_EXT.1.

2.6.3.2 Guidance Assurance Activities

The assurance activity for this requirement is performed in conjunction with the assurance activity for FMT_SMF_EXT.1.

2.6.3.3 Test Activities

The assurance activity for this requirement is performed in conjunction with the assurance activity for FMT_SMF_EXT.1.

2.7 Trusted Path/Channels (FTP)

2.7.1 Extended: Trusted channel Communication (FTP_ITC_EXT.1)

2.7.1.1 TSS Assurance Activity

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to access points, VPN Gateways, and other trusted IT products in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specifications. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

[ST] section 6.2.7.1.1 Network Protocols lists the trusted channel protocols that protect data in transit from disclosure, provide data integrity, and endpoint identification as TLS and HTTPS. [ST] section 6.8 Trusted Path / Channels states that: "Windows implements IEEE 802.11-2012, IEEE 802.1X and EAP-TLS to provide authenticated wireless networking sessions when requested by the user." [ST] section 6.4.3 states that Windows uses X.509 certificates for EAP-TLS exchanges, TLS, HTTPS, code signing for system software updates, and code signing for mobile applications..

If OTA updates are selected, the TSS shall describe which trusted channel protocol is initiated by the TOE and is used for updates.

"OTA updates" is not selected and this activity is not applicable.

2.7.1.2 Guidance Assurance Activities

The evaluator shall confirm that the operational guidance contains instructions for establishing the connection to access points, VPN Gateways, and other trusted IT products.

[Mobile Guide] section 4 Managing EAP-TLS describes configuration of security policy for wireless networks, both by an MDM[23] system (section 4.1 IT Administrator Guidance) and by a local Windows 10 administrator (section 4.2.1 Local Administrator Guidance).

[Mobile Guide] section 5 Managing TLS lists the cipher suites the TOE supports. The section covers configuring TLS via MDM[24] and for Windows 10 as a user and local administrator. The section includes links for Windows 10 to guidance to configure a server to allow only the specified cipher suites.

[Mobile Guide] Section 8 is Managing VPN. Subsection 8.1 IT Administrator's Guidance describes that an MDM system may be used to administer VPN profiles. The Windows IPsec VPN client can be configured by the MDM IT administrator, when the device is enrolled. The evaluated configuration requires that all network traffic other than traffic necessary to establish the VPN connection go through the VPN tunnel. This is done by verifying that the VPN configuration pushed down by the MDM is configured to "Send all traffic through the VPN connection".

[Mobile Guide] subsection 8.2 Windows 10 provides a TechNet topic that describes how to create a VPN connection. The Add-VpnConnection and Set-VpnConnection topic cover configuration to prevent split tunneling.

[Mobile Guide] section 13 Managing Certificates covers trusted channel policy. Subsection 13.1 IT Administrator Guidance describes setting Wi-Fi, VPN, and certificate profiles using an MDM as well as providing links to online guidance. Section 13.2 Windows 10 provides the same information for Windows 10 local administrators.

2.7.1.3 Test Activities

The evaluator shall also perform the following tests for each protocol listed:

Test 1: *The evaluators shall ensure that the TOE is able to initiate communications with an access point using 802.11-2012 and a pre-shared key, setting up the connections as described in the operational guidance and ensuring that communication is successful.*

This activity was performed in conjunction with FCS_CKM.1(2).

Test 2: *The evaluators shall ensure that the TOE is able to initiate communications with an access*

[23] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

[24] As indicated in section 1.1.2 Mobile Device management Solutions, [Mobile Guide] does not include specific steps for MDM solutions. Section 1.1.2 does include a reference to online information about supported MDM policies.

point using 802.11-2012, 802.1x, and EAP-TLS, setting up the connections as described in the operational guidance and ensuring that communication is successful.

This activity was performed in conjunction with FCS_TLSC_EXT.1.

Test 3: [conditional] *If IPsec is selected (and the TSF includes a native VPN client), the evaluator shall ensure that the TOE is able to initiate communications with a VPN Gateway, setting up the connections as described in the operational guidance and ensuring that communication is successful.*

This activity was performed in conjunction with FDP_IFC_EXT.1.

Test 4: *For any other selected protocol (not tested in Test 1, 2, or 3), the evaluator shall ensure that the TOE is able to initiate communications with a trusted IT product using the protocol, setting up the connection as described in the operational guidance and ensuring that the communication is successful.*

N/A—no other protocols are selected in the Security Target.

Test 5: *If OTA updates are selected, the evaluator shall trigger an update request according to the operational guidance and shall ensure that the communication is successful.*

The evaluator triggered an over the air update onto the TOE and verified that the TOE successfully communicated and received the update.

Test 6: *The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data are not sent in plaintext and that a protocol analyzer identifies the traffic as the protocol under testing.*

This activity was performed in conjunction with FDP_IFC_EXT.1, FCS_CKM.1(1), and FCS_TLSC_EXT.1.

3 Security Assurance Requirements

3.1 Class ADV: Development

3.1.1 ADV_FSP.1 Basic Functional Specification

3.1.1.1 FSP_FSP.1 Assurance Activity

There are no specific assurance activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the assurance activities

described in Section 5, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because the there is insufficient interface information, then an adequate functional specification has not been provided.

3.2 Class AGD: Guidance Documents

3.2.1 AGD_OPE.1 Operational User Guidance

3.2.1.1 AGD_OPE.1 Assurance Activity

Some of the contents of the operational guidance will be verified by the assurance activities in Section 5 and evaluation of the TOE according to the CEM. The following additional information is also required.

The operational guidance shall contain a list of natively installed applications and any relevant version numbers. If any third-parties vendors are permitted to install applications before purchase by the end user or enterprise, these applications shall also be listed.

[Mobile Guide] and [ST] refer to online Windows documentation, which was used for the assurance activities. All referenced online documentation applies to Windows mobile devices. See [Mobile Guide] section 1.1.1 Evaluated Configuration and [ST] section 2 TOE Description.

[Mobile Guide] section 30 includes a list of natively installed applications and indicates that they are Windows 10 version 10.0.10240.16384 and Windows 10 Mobile version 10586.13053.20151029.-1700.

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.

[Mobile Guide] section 21 Managing Cryptographic Algorithms state the Windows 10 system cryptographic engines were tested during the FIPS evaluation of the operating system. Other cryptographic engines may have been separately evaluated but were not part of this CC evaluation.

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

46. 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).

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

[Mobile Guide] Section 18 Managing Updates describe the process to update the TOE both by a local Windows 10 administrator (section 18.1.1 Local Administrator Guidance) and by a Windows 10 Mobile user (section 18.2.1 User Guidance).

Windows applications include metadata that is installed with the application by the Windows Installer and the Store App installer. The application metadata includes version information that prevents the

Windows Installer and the Store App installer from updating an installed application with an older version.

Update packages downloaded by Windows are signed with the Microsoft Root Certificate Authority to prove their authenticity and integrity. This signature is checked on the mobile device before installing any of the product updates contained in a given package in order to verify the updates have not been altered since they were digitally signed. If the signature is incorrect, then the update operation will fail. Otherwise, if the signature is correct then the update operation will proceed. Section 19 includes links to online documentation that describes how to determine whether an update operation was successful or unsuccessful.

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.

[Mobile Guide] states "This document provides operational guidance information for a Common Criteria evaluation describing only the security functionality which the administrator should use – any security functionality not described in this document is not part of the evaluation."

3.2.2 AGD_PRE.1 Preparative Procedures

3.2.2.1 AGD_PRE.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.

The evaluation team has reviewed the guidance documentation above and performed analysis of the guidance assurance activities for each applicable security functional requirement. The Assurance Activities for each SFR ensures that, where applicable, the guidance documentation provides adequate steps for a user to perform an action. This information is present within each section and for each platform claimed in the ST.

While performing testing, the evaluation team received the TOE and supported hardware and performed an installation of the test environment consistent with the evaluated configuration.

3.3 Class ALC: Life-Cycle Support

3.3.1 ALC_CMC.1 Labeling of the TOE Assurance Activity

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

The evaluator reviewed the configuration of the TOE during testing and confirmed that Windows 10 or Windows 10 Mobile was the tested version for each platform in the evaluation. All versions were claimed within the ST and all documentation uniquely identified the TOE version as Windows 10 or Windows 10 Mobile.

3.3.2 ALC_CMS.1 TOE CM Coverage Assurance Activity

The evaluator shall ensure that the developer has identified (in public-facing development documentation for their 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.

In regards to developer environments, developers who wish to develop for the devices can navigate to http://dev.windows.com/en-us/ and follow the necessary procedures. Before submitting a developed application, an individual must have a developer account and must meet all guidelines. Once submitted, it would be reviewed.

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 boundary is identified within the Security Target and the guidance documentation clearly identifies how a TOE user would place the product in the evaluated configuration.

3.3.3 Timely Security Updates (ALC_TSU_EXT) 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. The evaluator shall verify that this description addresses the TOE OS, the firmware, and bundled applications, each. The evaluator shall also verify that, in addition to the TOE developer's process, any carrier or other third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

[ST] section 6.6.6 Windows and Application Updates describes the process of creating and signing a security update for the TOE. The ST also describes how the TOE validates a security updated along with how a user can receive automatic updates or obtain updates manually. Section 6.9 Security Response Process states the processes in section 6.6.6 apply to the operating system, firmware, and applications.

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. The evaluator shall verify that this time is expressed in a number or range of days.

[ST] claims timely action without identifying a specific time between public disclosure and update availability. Rather, section 6.9 Security Response Process provides links to pages describing

Microsoft's Security Update Lifecycle along with Report a Computer Security Vulnerability and Microsoft Security Response Policy and Practices. The life cycle includes monthly security bulletins and updates.

The evaluator shall verify that this description includes the publically available mechanisms (including either an email address or website) for reporting security issues related to the TOE. 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 6.9 Security Response Process provides an email address and web page (HTTPS link) for reporting security issues. The section provides a secure link (HTTPS) to Microsoft Security Response Center PGP Key and S/MIME certificate for securing email communication.

3.4 ATE_IND.1 Independent Testing Conformance

3.4.1 ATE_IND.1 Assurance Activity

The evaluator shall prepare a test plan and report documenting the testing aspects of the system. The test plan covers all of the testing actions contained in the CEM and the body of this PP's Assurance Activities. While it is not necessary to have one test case per test listed in an Assurance Activity, the 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 affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the 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. 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 (IPsec, TLS/HTTPS, 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. 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 plan created for testing of Windows 10 and Windows 10 Mobile was performed against the requirements of the Mobility Device Fundamentals Protection Profile 2.0. Each test case is mapped to a requirement which is met with a passing result. For each case, a description, expected result, actual result, and evidence are provided to clearly identify how the requirement was met. Testing was performed by analyzing the information within the AGD to ensure the evaluator could follow guidance procedures in order to complete the configuration activities required. For certain tests, vendor apps were needed to test the product. These actions would not normally be performed by a TOE user and would only occur during testing for [PP MDF]. The overall conclusion was that testing was successful for all requirements.

3.4.2 Cryptographic Algorithm Validation Programming Testing

Windows 10 and Window 10 Mobile use algorithm implementations validated under the Cryptographic Algorithm Validation Program (http://csrc.nist.gov/groups/STM/cavp/index.html). [ST] Table 14 Cryptographic Algorithm Standards and Evaluation Methods identifies CAVP certificates that apply to Windows 10 and Windows 10 Mobile devices. This section clarifies correspondence between operational environments listed on CAVP certificates and TOE mobile devices. Information on individual tests, modes, states, and key sizes can be found in cryptographic requirement sections above:

- 2.1.1.3
- 2.1.1.5
- 2.1.4.3
- 2.1.4.6
- 2.1.12.3
- 2.1.13.3
- 2.1.14.3
- 2.1.15.3
- 2.1.19.3

Windows 10 and Window 10 Mobile run in a large set of operational environments. CAVP certificates cover a range of Windows variants, architectures, and cryptographic feature implementations (for example, AES-NI). The correspondence between CAVP test systems and the Windows 10 and Windows 10 Mobile device undergoing CC evaluation is not one-to-one and is not required to be. *NIAP Policy Letter #5* (https://www.niap-ccevs.org/Documents_and_Guidance/ccevs/policy-ltr-5-update1.pdf) defines the applicability and relationship of NIST CAVP and CMVP testing to assurance activities associated with cryptography requirements in NIAP Protection Profiles. NIAP relies on NIST to establish guidance and to determine when operational environments are equivalent. *Frequently Asked Questions for NIAP Policy #5* provides guidance on applying CAVP certificates to assurance activities in NIAP protection profiles (https://www.niap-ccevs.org/Documents_and_Guidance/ccevs/FAQ_Policy_5.pdf). This guidance is based on *Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program* (IG: http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-2/FIPS1402IG.pdf).

Policy 5 FAQ 4 addresses equivalence of operational environments. FAQ 4 explains:

> If the untested processor supports the same instruction set and operates on the same word size as the tested processor and the algorithm implementation can operate on the untested processor without change, then the algorithm implementation **does not have to be re-tested.**

[ST] identifies the processor for each mobile devices in section 6.6.1.1.1 Processor and Memory

Device	Processor	Instruction Set	Word Size
Microsoft Surface Pro 4	Intel Core i7	x64	64-bit
Microsoft Lumia 950	Snapdragon 808 Hexacore	ARMv8-A	64-bit/32-bit (Cortex-A57/Cortex-A53)
Microsoft Lumia 950 XL	Snapdragon 810 Octacore	ARMv8-A	64-bit/32-bit (Cortex-A57/Cortex-A53)
Microsoft Lumia 550	Snapdragon 210	ARMv7	32-bit (Cortex-A7)
Microsoft Lumia 635	Snapdragon 400 Quad-core ARM Cortex A7	ARMv7	32-bit (Cortex-A7)

Windows 10 runs without change on Intel Core i7 and Intel Core i5 processors. Core i7 and Core i5 processors are 64-bit processors that support the same x64 instruction sets including AES-NI, PCKMULQDQ, and SSSE 3 extensions. Thus, the following operational environments cited on CAVP certificates apply to the Surface Pro 4 mobile device:

- Intel Core i7 with AES-NI w/ Microsoft Surface Pro 3 w/ Windows 10 Enterprise (x64)
- Intel Core i7 with AES-NI w/ Microsoft Surface Pro 3 w/ Windows 10 Pro (x64)
- Intel Core i5 with AES-NI w/ Microsoft Surface Pro 2 w/ Windows 10 Enterprise (x64)
- Intel Core i5 with AES-NI w/ Microsoft Surface Pro 2 w/ Windows 10 Pro (x64)

The CAVP certificates include additional equivalent operational environments (that is, 64-bit x64 Windows 10 that runs unmodified) with Intel and non-Intel processors.

- Intel x64 Processor with AES-NI w/ Microsoft Surface Pro w/ Windows 10 Enterprise (x64)
- Intel x64 Processor with AES-NI w/ Microsoft Surface Pro w/ Windows 10 Pro (x64)
- AMD A4 with AES-NI and PCLMULQDQ and SSSE 3 w/ Windows 10 Enterprise (x64)
- AMD A4 with AES-NI and PCLMULQDQ and SSSE 3 w/ Windows 10 Pro (x64)

There was no technical reason to differentiate between Windows 10 Pro and Enterprise. Microsoft uses the same binaries that implement cryptography in Windows 10 Pro and Windows 10 Enterprise editions. The testing of additional Operational Environments supports the assessment that there are no technical difference between Windows 10 Enterprise and Windows 10 Pro.

Consequently, the CAVP certificates for Windows 10 cited in the security target apply to the Surface Pro 4 mobile device.

Similarly, the Windows 10 November 2015 Update CAVP certificates cited for Microsoft Lumia 950 apply to Microsoft Lumia 950 XL. The CAVP certificates for Microsoft Lumia 635 apply to Microsoft Lumia 550. The applicable environments are:

- Qualcomm Snapdragon 808 (A57, A53) w/ Microsoft Lumia 950
- Qualcomm Snapdragon 400 (A7) w/ Microsoft Lumia 635 w/ Windows 10 Mobile

3.5 Class AVA: Vulnerability Assessment

3.5.1 AVA_VAN.1 Assurance Activity

As with ATE_IND, the evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to determine the vulnerabilities that have been found in network infrastructure devices and the implemented communication protocols in general, as well as those that pertain to the particular TOE. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. For example, if the vulnerability can be detected by pressing a key combination on boot-up, a test would be suitable at the assurance level of this PP. 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.

The evaluation team applied the Vulnerability Analysis approach above to the Windows 10 and Windows 10 Mobile TOE. The team documented the analysis and results Vulnerability Analysis Report, which the team provided to the Common Criteria Evaluation and Validation Scheme certification body.