www.GossamerSec.com

Assurance Activity Report for SecuSuite V5.0 and SteelBox V5.0

Version 0.4
12/08/22

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Revision History

Revision	Date	Authors	Summary
Version 0.1	11/16/22	Khai Van	Initial draft
Version 0.2	12/05/22	Khai Van	Updated according to NIAP comments
Version 0.3	12/06/22	Khai Van	Updated with VVoIP module FPT_TUD evidence
Version 0.4	12/08/22	Khai Van	Updated TOE name

The TOE Evaluation was Sponsored by:

BlackBerry Ltd.

Evaluation Personnel:

- Khai Van
- Matai Spivey

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction

This document presents evaluations results of the SecuSUITE v5.0 and SteelBox v5.0 ASPP14/VVoIPAS10/PKGTLS11 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 References

The following evidence was used to complete the Assurance Activities:

- Samsung Galaxy Devices on Android 11 - Spring [VID11160 ST]**:** https://www.niap-ccevs.org/Product/Compliant.cfm?PID=11160
- Apple iOS 14: iPhones [VID11146 ST]: https://www.niap-ccevs.org/product/Compliant.cfm?PID=11146
- SecuSUITE v5.0 and SteelBox v5.0 Security Target, Version 0.6, 12/07/2022 [ST]
- Common Criteria Configuration Guide SecuSUITE v5.0 SteelBox v5.0, Version 1.1, 05-Dec-2022 [Admin Guide]

The following is the relevant PP configuration and components used for this evaluation:

- PP-Configuration for Application Software and Voice/Video over IP (VVoIP) Endpoints, Version 1.1, 31 May 2022
  - Protection Profile for Application Software, Version 1.4, 2021-10-07 (PP_APP_V1.4)
  - PP-Module for Voice/Video over IP (VVoIP) Endpoints, Version 1.0, 28 October 2020 (MOD_VVoIP_V1.0)
- Functional Package for Transport Layer Security (TLS), 1.1, 2019-02-12 (PKGTLS11)

Note that in the AAs below, the SFRs beginning with VVOIPAS10: belong to the ASPP SFRs in VVOIPAS10.

1.2 TOE CAVP Certificates

The following table shows CAVP certificates for each algorithm:

SFR	Algorithm	Certificate Number
ASPP14:FCS_CKM.1 ASPP14:FCS_CKM.1/AK	FIPS 186-4 ECDSA Key Generation P-256, 384	A2639
ASPP14:FCS_CKM.2	SP 800-56Ar3 Elliptic curve-based key establishment schemes	A2639
ASPP14:FCS_COP.1/Hash VVoIPAS10:FCS_COP.1/SRTP	SHS Hash sizes supported: SHA-1, SHA-256, SHA-384, SHA-512	A2639
ASPP14:FCS_COP.1/KeyedHash VVoIPAS10:FCS_COP.1/SRTP	HMAC Hash sizes supported: SHA-1, SHA2-256, SHA2-384	A2639

SFR	Algorithm	Certificate Number
ASPP14:FCS_COP.1/Sig	FIPS 186-4 Elliptic Curve Digital Signature Algorithm (ECDSA) with an elliptical curve size of 256 or 384 bits FIPS 186-4 RSA Digital Signature with 2048 or 3072 bit keys (verification only)	A2639
ASPP14:FCS_COP.1/SKC	AES-GCM Encrypt/Decrypt Key sizes supported: 256 bits	A2639
VVoIPAS10:FCS_COP.1/SRTP	AES-CTR (256 bit) AES-GCM (256-bit)	A2639
ASPP14:FCS_RBG_EXT.2	SP 800-90A CTR_DRBG with sw based noise sources with a minimum of 256 bits of non-determinism	A2639

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profile and describes the findings in each case.

2.1 Communication (FCO)

2.1.1 Fixed-Rate Vocoder (VVoIPAS10:FCO_VOC_EXT.1)

2.1.1.1 VVoIPAS10:FCO_VOC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS specifies each vocoder used. The evaluator shall then examine the specification for each vocoder in order to verify that no variable rate vocoders are claimed by the TSF.

VVoIPAS10:FCO_VOC_EXT.1 in the [ST] states that the TOE uses the Opus codec by default, which utilizes a fixed bit-rate value. The TOE also includes the SILK codec as a secondary codec for backwards compatibility with older versions of the TOE. SILK is a variable bit-rate codec by default, however the TOE utilizes a custom build that includes a bit-rate padding to make the codec a fixed bit-rate value. The SILK codec is only used if the peer device doesn't support the Opus codec.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall set up a test environment that contains the TOE, a call control server (which may be the TOE itself), a network switch, a packet capture tool, and a second VVoIP endpoint.

The evaluator shall then perform the following test:

1. The evaluator shall ensure the TOE and the second VVoIP endpoint are registered to a call control server or are using P2P.

2. The evaluator shall use the TOE to dial the second VVoIP endpoint to establish a call and verify the call is established by holding a voice conversation.

3. The evaluator shall review the captured traffic to verify that a fixed rate vocoder is used.

If multiple vocoders are supported, the evaluator shall reconfigure the TOE to use each individual vocoder and repeat steps 1-3 for each vocoder.

The evaluator registered the TOE and a second VoIP client running SecuSUITE version 5.0 with a SecuGATE call control server. The evaluator established a call between the TOE and the second endpoint. While the call is ongoing, the evaluator enabled SIP packet capturing on the SecuGATE server. The evaluator reviewed the packet capture and determined that the call session was established using the Opus codec. The evaluator repeated this test by establishing a call between the TOE and a VoIP client running SecuSUITE version 4.0. The evaluator reviewed the packet capture and observed that the SILK codec was used. In both cases, the evaluator reviewed the media data between the TOE and the endpoint. The evaluator concluded that as the payload lengths remained the same for all of the media packets, the codecs were indeed operating in a constant bit-rate.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation Services (ASPP14:FCS_CKM.1)

2.2.1.1 ASPP14:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The evaluator examined ASPP14:FCS_CKM.1 in the [ST] and determined that the TOE needs asymmetric key generation services. The TOE invokes platform-provided key generation services on an Apple iOS device. The TOE implements its own asymmetric key generation on an Android device.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.2 Cryptographic Asymmetric Key Generation - per TD0659 (ASPP14:FCS_CKM.1/AK)

2.2.2.1 ASPP14:FCS_CKM.1.1/AK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

The "Cryptographic support" section of the [ST] states that the TOE generates TLS key exchange keys using P-384. The TOE also supports generating P-256 keys for backwards compatibility with older Blackberry SecuGATE SIP servers. On Android, the TOE uses the internal cryptographic module's asymmetric key generation services to generate asymmetric keys. On iOS, the TOE invokes platform provided functionality to generate asymmetric keys. The TOE uses Security Framework from iOS, specifically the APIs in /System/Library/Frameworks/Security.framework/Security.

Component 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 “General Common Criteria Configuration” of the [Admin Guide] states that cryptographic keys used for the generation of client TLS certificates are created based on the client configuration submitted by the BlackBerry SecuGATE server (ESC) during initial client enrollment. The configuration is defaulted in SecuGATE Server to the usage of NIST p-384 curve and cannot be changed.

Component Testing Assurance Activities: If the application 'implements asymmetric key generation,' then the following test activities shall be carried out. Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application. Key Generation for FIPS PUB 186-4 RSA Schemes The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e,

the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:

Provable primes

Probable primes

2. Primes with Conditions:

Primes p1, p2, q1,q2, p and q shall all be provable primes

Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

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

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

n = p*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 >= 2^(nlen/2 -1/2),

q >= 2^(nlen/2 -1/2),

2^(nlen/2) < d < LCM(p-1,q-1),

e*d = 1 mod LCM(p-1,q-1).

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P384 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., P256, 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. 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 not= 0,1

q divides p-1

g^q mod p = 1

g^x mod p = y

for each FFC parameter set and key pair.

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

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.3 Cryptographic Key Establishment (ASPP14:FCS_CKM.2)

2.2.3.1 ASPP14:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The "Cryptographic support" section of the [ST] states that the TOE performs ECDHE key establishment schemes, which correspond to the ST's ASPP14:FCS_CKM.1/AK claim that the TOE supports elliptic curve-based key establishment schemes that meets NIST Special Publication 800-56A. The TOE uses the key establishment in TLS key exchange.

Component 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).

Section “General Common Criteria Configuration” of the [**Admin Guide**] states that the only key establishment scheme is ECDHE.

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

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the OtherInfo field, the

data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

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

SP800-56B Key Establishment Schemes

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

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

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if 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 evaluation activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

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

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the

contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

RSA-based key establishment

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

Diffie-Hellman Group 14

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

FFC Schemes using 'safe-prime' groups

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.4 Cryptographic Operation - Hashing (ASPP14:FCS_COP.1/Hash)

2.2.4.1 ASPP14:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section "Cryptographic support" in the [ST] states that the TOE uses SHA-256 as the SIP message digest authentication mechanism during a SIP transaction. All SIP messages are protected via TLS. The TOE supports SHA-

1 in the SRTP ciphersuite. The TOE supports both SHA-256 and SHA-384 in digital signatures. The TOE supports SHA-384 and SHA-512 in the TLS signature algorithms extension.

Component Guidance Assurance Activities: None Defined

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

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

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

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

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

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

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.5 Cryptographic Operation - Keyed-Hash Message Authentication - per TD0626 (ASPP14:FCS_COP.1/KeyedHash)

2.2.5.1 ASPP14:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the "TOE CAVP Certificates" table.

2.2.6 Cryptographic Operation - Signing (ASPP14:FCS_COP.1/Sig)

2.2.6.1 ASPP14:FCS_COP.1.1/Sig

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following activities based on the selections in the ST.

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

ECDSA Algorithm Tests

Test 1: ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

Test 2: ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Test 1: Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Test 2: Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.7 Cryptographic Operation - Encryption/Decryption (ASPP14:FCS_COP.1/SKC)

2.2.7.1 ASPP14:FCS_COP.1.1/SKC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component 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 modes and key sizes is present.

Section "TLS configuration" in the [**Admin Guide**] states that the TOE supports two cipher suites, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 and TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384. The cipher suite is selected by the SecuGATE server during TLS handshake and currently defaulted to TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. This ciphersuite can be changed in the SecuGATE.

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

AES-CBC Known Answer Tests

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

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

PT = IV

else:

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

PT = CT[i-1]

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Monte Carlo Tests

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

128 bit and 256 bit keys

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

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

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

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

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

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

AES-XTS Tests

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

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

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

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt. The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

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

AES-CCM Tests

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

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

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

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

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

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

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

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

Variable Associated Data Test

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

Variable Payload Test

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

Variable Nonce Test

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

Variable Tag Test

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

Decryption-Verification Process Test

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

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

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

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

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

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

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

functionality, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input.

Test 2: Multi-Block Message Test

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

Test 3: Monte-Carlo Test

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

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

For AES-ECB mode

Input: PT, Key

for i = 1 to 1000:

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

PT = CT[i]

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.8 Cryptographic Operation (Encryption/Decryption for SRTP) (VVoIPAS10:FCS_COP.1/SRTP)

2.2.8.1 VVoIPAS10:FCS_COP.1.1/SRTP

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine whether it claims the use of GCM. If it does, the evaluator shall verify that the TSS notes the GCM key sizes that the TOE supports for each use of GCM. If GCM is supported for other uses, the evaluator shall ensure that the specific key sizes supported for each use of GCM are identified.

Section "Cryptographic support" in the [ST] states that the TOE uses AES-GCM in the SRTP ciphersuite as well as the encryption algorithm in the TLS ciphersuites. In both instances, the TOE uses AES-GCM with 256-bit keys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

AES-CTR Tests (Conditional - If 'AES-CTR (as defined in NIST SP 800-38A)' is selected in FCS_COP.1.1/SRTP):

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Since the Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AESGCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

AES-CTR Known Answer Tests

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

KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 5 plaintext values for each selected keysize and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros.

KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 5 key values for each selected keysize and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value.

KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the

given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 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].

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte Carlo Test

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

Input: PT, Key

for I = 1 to 1000:

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

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

There is no need to test the decryption engine.

AES-GCM Tests (Conditional - If 'AES-GCM (as defined in NIST SP 800-38D)' is selected in FCS_COP.1.1/SRTP):

If the TOE claims the App PP as its Base-PP, the evaluator shall perform the AES-GCM test activities identified for FCS_COP.1(1) in the Base-PP.

If the TOE claims the NDcPP as its Base-PP, the evaluator shall perform the AES-GCM test activities identified for FCS_COP.1/DataEncryption in the Base-PP.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.9 HTTPS Protocol (ASPP14:FCS_HTTPS_EXT.1/Client)

2.2.9.1 ASPP14:FCS_HTTPS_EXT.1.1/Client

TSS Assurance Activities: The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Section "Cryptographic support" in the [ST] states that the TOE conforms to RFC 2818 by securing HTTPS with TLSv1.2. The TOE uses TLS to protect communications with SecuGATE for all activities such as client configuration and certificate enrollment. The TOE's TLS implementation has been tested in accordance with the activities in the Functional Package for Transport Layer Security (TLS), version 1.1.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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.

The TLS testing performed in the assurance activities of the Functional Package for Transport Layer Security (TLS), version 1.1, adequately show that the TOE's communications with the webserver are secured. The evaluator inspected the packet captures from the TLS testing and determined that the TOE secures the channel with the webserver using TLS.

2.2.9.2 ASPP14:FCS_HTTPS_EXT.1.2/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Other tests are performed in conjunction with the TLS package.

These tests are performed in conjunction with the TLS package.

2.2.9.3 ASPP14:FCS_HTTPS_EXT.1.3/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and 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 selected action in the SFR. If 'notify the user' is selected in the SFR, then the evaluator shall also determine that the user is notified of the certificate validation failure. 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 again, using a certificate without a valid certification path results in the selected action in the SFR, and if 'notify the user' was selected in the SFR, the user is notified of the validation failure.

This test is performed in PKGTLS11:FCS_TLSC_EXT.1.3 tests 1a and 1b, where the TOE demonstrates validation of the certification path and responds accordingly to either a valid or invalid path.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.10 Random Bit Generation Services (ASPP14:FCS_RBG_EXT.1)

2.2.10.1 ASPP14:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

If 'invoke platform-provided DRBG functionality' is selected, the evaluator performs the following activities.

The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these

functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

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

The [**ST**] claim for ASPP14:FCS_RBG_EXT.1 is that the TOE implements DRBG functionality. The evaluator made sure that the [**ST**] also includes FCS_RBG_EXT.2 elements.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: If 'invoke platform-provided DRBG functionality' is selected, the following tests shall be performed:

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

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

Platforms: Android....

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

Platforms: Microsoft Windows....

The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

Platforms: Apple macOS....

The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

This test is not applicable as the TOE implements DRBG functionality.

2.2.11 Random Bit Generation from Application (ASPP14:FCS_RBG_EXT.2)

2.2.11.1 ASPP14:FCS_RBG_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests, depending on the standard to which the RBG conforms. Implementations Conforming to FIPS 140-2 Annex C. The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the 'expected values' are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

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

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

Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

Test 1: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

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

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 less then or equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in the “TOE CAVP Certificates” table.

2.2.11.2 ASPP14:FCS_RBG_EXT.2.2

TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

The TOE's [**Entropy Report**] contains the justifications for the TOE's deterministic random bit generator in terms of its design, entropy output quality, operating conditions, and health testing.

Guidance Assurance Activities: None Defined

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

No current test.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.12 Secure Real-Time Transport Protocol (VVoIPAS10:FCS_SRTP_EXT.1)

2.2.12.1 VVoIPAS10:FCS_SRTP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 VVoIPAS10:FCS_SRTP_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.3 VVoIPAS10:FCS_SRTP_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.4 VVoIPAS10:FCS_SRTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes how the SRTP session is negotiated for both incoming and outgoing calls. This includes how the keying material is established, as well has how requests to use the NULL algorithm or other unallowed cipher suites are rejected by the TSF.

Section "Cryptographic support" of the [**ST**] states that the TOE supports SRTP protocol as described by RFC 3711 using Security Descriptions for Media Streams (SDES) in compliance with RFC 4568. The SRTP session is negotiated via the SecuGATE SIP server (also known as the ESC). The TOE supports both AEAD_AES_256_GCM and AES_256_CM_HMAC_SHA1_80 ciphersuites for SDES-SRTP. The TOE and its VoIP peer must be registered to an ESC (SecuGATE) in order to communicate with each other. The TOE does not allow the NULL algorithm to be specified and will reject connections from an endpoint that use any invalid algorithm, including the NULL ciphersuite. The SRTP destination ports that shall be used for a specific call are 'negotiated' between RTP Proxy and SIP Server and included into the SDES body of the forwarded SIP messages. The admin can restrict the port range in the SIP server configuration.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it includes instructions for how to disable the use of the SRTP NULL algorithm and how to specify the ports to be used for SRTP communications.

Section "General Common Criteria Configuration" in the [**Admin Guide**] states that the ports used for exchanging RTP packets are defined by the SecuGATE during call establishment. The SRTP NULL algorithm is disabled in the TOE and no configuration is necessary to disable the NULL cipher.

Component Testing Assurance Activities: The evaluator shall follow the procedure for initializing their device so that they are ready to receive and place calls. For each cipher suite selected in FCS_SRTP_EXT.1.2, the evaluator shall configure the SIP server to only allow that cipher suite to be used. The evaluator shall then both place and receive a call and determine that the traffic sent and received by the TOE is encrypted using SRTP with that cipher suite. The evaluator may choose one of the below two options to ensure that the call is being encrypted and to view the cipher suite being used.

Option 1: The evaluator shall configure the SIP server to report whether SRTP is being used, and if so, print the negotiated SRTP cipher suite. The evaluator shall confirm that SRTP was used for the calls and that the correct cipher suite was negotiated.

Option 2: A packet capture tool should be used with the SIP server's private key loaded in. The evaluator shall decrypt the TLS-SIP traffic, view the SDES negotiation, and ensure that the correct cipher suite was negotiated.

Next, the evaluator shall configure the SIP server to only allow the SRTP NULL cipher suite. The evaluator shall attempt to both place and receive a call and confirm that both attempts failed.

The TOE uses a specific ciphersuite based on the SecuSUITE version of the VoIP peer. If the peer's SecuSUITE version is 5.0, then the TOE uses the AEAD_AES_256_GCM ciphersuite. If the peer's SecuSUITE version is 4.0 or lower, then the TOE uses the AES_256_CM_HMAC_SHA1_80 ciphersuite.

The evaluator registered the TOE on both the Android and iOS platforms to the SIP server. The evaluator also registered another VoIP endpoint that alternates between SecuSUITE versions 4.0 and 5.0.

The evaluator then placed a call between the TOE and the second endpoint, alternating between the two different SecuSUITE versions on the second endpoint. The evaluator verified from the TOE's debug logs that the TOE (on both Android and iOS platforms) uses SRTP and is able to negotiate the claimed cipher suite based on the peer's SecuSUITE version.

In order to attempt a negotiation of the NULL ciphersuite, the evaluator used a special build of the SecuGATE server that forces the use of the NULL ciphersuite. The evaluator also set up a breakout call environment by setting up two SecuGATES which are configured to perform SIP trunking with each other. This setup ensures that the NULL ciphersuite is the only ciphersuite offered to the TOE. The evaluator registered the TOE (both Android and iOS) to the first ESC and a secondary endpoint to the second ESC. The evaluator then placed a call between the TOE and the second endpoint. The evaluator observed the TOE rejecting the call connections. The TOE's debug logs on each platform show that the TOE rejected the attempt at a negotiation using the NULL ciphersuite.

2.2.13 Storage of Credentials (ASPP14:FCS_STO_EXT.1)

2.2.13.1 ASPP14:FCS_STO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

According to section 6.2 of the [**ST**], the TOE stores X509v3 certificate private keys and the SRTP encryption key persistently. The TOE uses X509 certificates to authenticate to SecuGATE when performing TLS. The TOE uses an AES key to encrypt SRTP media during a call session. The TOE stores the data when not in use in client device platform-provided key storage as follows:

- Samsung: Android KeyStore
- Apple: iOS keychain

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/SKC or conditioned according to FCS_CKM.1.1/AK and FCS_CKM.1/PBKDF. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

The evaluator shall verify that all credentials are stored within Keychain

The evaluator determined that the TOE on Android uses the Android KeyStore by observing the use of the Android keystore file format in the Android internal filesystem. For iOS, the evaluator searched a decompiled application for instances of the iOS keychain and found that the TOE uses several API calls for manipulating the iOS key chain files.

2.2.14 TLS Protocol (PKGTLS11:FCS_TLS_EXT.1)

2.2.14.1 PKGTLS11:FCS_TLS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Section 5 of the [**Admin Guide**] refers to the TOE as a client, and the TOE acts as a TLS client when it connects to the ESC. The TLSC selections in the [**ST**] are consistent with the TOE's behavior.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.15 TLS Client Protocol (PKGTLS11:FCS_TLSC_EXT.1)

2.2.15.1 PKGTLS11:FCS_TLSC_EXT.1.1

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

Section "Cryptographic support" of the [**ST**] specifies the TLS cipher suites supported by the TOE, which are the following:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Guidance Assurance Activities: The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS.

Section "General Common Criteria Configuration" of the [**Admin Guide**] states that the TOE offers only two ciphersuites to the SecuGATE server, which are TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 and TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384. The SecuGATE server selects the ciphersuite, which defaults to TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384.

Testing Assurance Activities: The evaluator shall also perform the following tests:

Test 1: The evaluator shall establish a TLS connection using each of the cipher suites 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 cipher suite 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 cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation. The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

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

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

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

Test 5.1: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

Test 5.2: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.

Test 5.3: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.

Test 5.4: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.

Test 5.5: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

Test 5.6: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.

Test 5.7: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

Test 1: The evaluator attempted to establish a TLS session using each of the cipher suites specified by the requirement. The evaluator observed that the TOE only accepts connections using the cipher suites claimed in the [ST].

Test 2: The evaluator connected the TOE to an ESC which uses a valid certificate in the TLS connection. The TOE successfully connects. The evaluator then connected the TOE to an ESC which uses a certificate missing the Server Authentication key usage. The TOE rejects the connection.

Test 3: The evaluator attempted to connect the TOE to an ESC that returns a server certificate in the TLS connection that does not match the server-selected cipher suite. The TOE rejects the connection.

Test 4: The evaluator connected the TOE to an ESC that attempted to use the NULL ciphersuite with TLS. The TOE rejects the connection.

Test 5.1: The evaluator connected the TOE to an ESC that reported an undefined TLS version. The TOE rejects the connection.

Test 5.2: The evaluator connected the TOE to an ESC that reported an unsupported recent TLS version. The TOE rejects the connection.

Test 5.3: The evaluator connected the TOE to an ESC with a TLS connection in which the server nonce was modified. The TOE rejects the connection.

Test 5.4: The evaluator connected the TOE to an ESC with a TLS connection in which the ciphersuite is not one matching a ciphersuite in the client hello. The TOE rejects the connection.

Test 5.5: The evaluator connected the TOE to an ESC with a TLS connection in which the server's key exchange signature block was modified. The TOE rejects the connection.

Test 5.6: The evaluator connected the TOE to an ESC with a TLS connection in which the server's ServerFinished message was modified. The TOE rejects the connection.

Test 5.7: The evaluator connected the TOE to an ESC with a TLS connection in which the server sent a garbled message after the Change Cipher Spec message. The TOE rejects the connection and no application data was present.

2.2.15.2 PKGTLS11:FCS_TLSC_EXT.1.2

TSS Assurance Activities: 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. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the product.

Section "Cryptographic support" in the [**ST**] states that the TOE supports TLS communication with mutual authentication as described above. The TOE uses the connection URL as the reference identifier to compare against the identifier in the peer's certificate. The TOE supports both CN and SAN reference identifier checking, with the SAN check preferred over CN. The TOE verifies the certificate received matches the reference identifier for the expected peer. The TOE does not accept certificates that cannot be determined to be valid. The TOE does not support pinned certificates and URIs. Wildcards are not supported.

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.

The "Activating the SecuSUITE app" section of the [**Admin Guide**] states that the TOE uses the URL entered by the user or derived from the QR code information as a reference identifier for the TLS certificate validation.

Testing Assurance Activities: The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned

certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7. (TD0499 applied)

Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

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: [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

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 support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.

Test 5.1: [conditional]: If wildcards are supported, 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.

Test 5.2: [conditional]: If wildcards are supported, 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.come) and verify that the connection fails.

Test 5.3: [conditional]: If wildcards are supported, 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 5.4: [conditional]: If wildcards are not supported, the evaluator shall present a server certificate containing a wildcard in the left-most label (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 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.

Test 1: The evaluator attempted to connect the TOE to an ESC in which the ESC's certificate contains a CN that does not match the reference identifier and does not contain the SAN extension. The TOE rejects the connection.

Test 2: The evaluator attempted to connect the TOE to an ESC in which the ESC's certificate contains a CN that matches the reference identifier, but also contains a SAN that does not match the reference identifier. The TOE rejects the connection.

Test 3: The evaluator attempted to connect the TOE to an ESC in which the ESC's certificate contains a CN that matches the reference identifier, but is missing a SAN. The TOE accepts the connection.

Test 4: The evaluator attempted to connect the TOE to an ESC in which the ESC's certificate contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The TOE accepts the connection.

Test 5 (5.1 through 5.4): The TOE does not support wildcards. The evaluator attempted several wildcard tests (both positive and negative test cases) and verified that the TOE rejects all certificates with wildcards.

Test 6: This test is not applicable as the TOE does not support URI or Service name reference identifiers.

Test 7: This test is not applicable as the TOE does not support pinned certificates.

2.2.15.3 PKGTLS11:FCS_TLSC_EXT.1.3

TSS Assurance Activities: If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.

The TOE rejects any invalid connection attempt with no exceptions.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows, unless excepted:

Test 1a: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.

Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.

Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.

(TD0513 applied)

Test 2: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.

Test 3: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.

Test 4: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

Test 1a: The evaluator attempted to connect the TOE to an ESC that returns a certificate signed by a CA recognized by the TOE. The TOE accepts the connection.

Test 1b: The evaluator attempted to connect the TOE to an ESC that returns a certificate chain missing an intermediate CA. The TOE rejects the connection.

Test 1c: This test is not applicable as the TOE trust store cannot by managed.

Test 2: The evaluator attempted to connect the TOE to an ESC that returns a revoked server certificate. The TOE rejects the connection.

Test 3: The evaluator attempted to connect the TOE to an ESC that returns an expired server certificate. The TOE rejects the connection.

Test 4: The reference identifier tests are performed in PKGTLS11:FCS_TLSC_EXT.1.2.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.16 TLS Client Support for Mutual Authentication (PKGTLS11:FCS_TLSC_EXT.2)

2.2.16.1 PKGTLS11:FCS_TLSC_EXT.2.1

TSS Assurance Activities: 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. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.

Section "Identification and authentication" in the [**ST**] contains a description of situations that the TOE uses certificates for TLS mutual authentication with a SecuGATE SIP server. The TOE utilizes OpenSSL version 1.0.2zf to perform TLS communications and X509 certificate checking. The TOE has multiple interfaces that use TLS:

SCA Registration (TCP port 3978): This channel is for the initial set up to register the TOE to the SecuGATE.

SIP communications (TCP port 5061): This channel is used for securing the media channel setup between the TOE and SecuGATE.

Secure client authentication (TCP port 5062): This channel is used in secure communications with SecuGATE for activities such as updating configuration and secure contacts.

In all cases, the TOE checks the TLS peer's certificate. The TOE only supports mutual authentication when it connects to SecuGATE for SIP communications and secure client authentication. Certificates are provisioned by the SecuGATE SIP Server during SCA registration. The TOE uses certificates automatically, and these certificates are exchanged during TLS negotiations for the media and configuration communications with SecuGATE. If the TOE determines the certificates from the peer are not valid, the certificates are not accepted and the TOE rejects the connection attempt. If an otherwise valid certificate cannot have its CRL checked to confirm that the certificate is not revoked, the certificate is not accepted and the TOE rejects the connection.

Guidance Assurance Activities: The evaluator shall ensure that the AGD guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also 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.

Section "General Common Criteria Configuration" of the [**Admin Guide**] states that X.509 certificates used for mutual TLS authentication are provisioned by the SecuGATE SIP server during the initial client registration process

executed between the TOE and the SecuGATE server.

Testing Assurance Activities: The evaluator shall also perform the following tests:

Test 1: The evaluator shall establish a connection to a server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

Test 2: The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.

Test 1: The evaluator configured the ESC server without mutual authentication and attempted to connect the TOE to the ESC. The evaluator verified that the TOE does not send the Client’s Certificate message (type 11) during the handshake.

Test 2: The evaluator configured the ESC server with mutual authentication and attempted to connect the TOE to the ESC. The evaluator verified that the TOE responds with a non-empty Client’s Certificate message (type 11) and Certificate Verify (type 15) message.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.17 TLS Client Support for Signature Algorithms Extension (PKGTLS11:FCS_TLSC_EXT.3)

2.2.17.1 PKGTLS11:FCS_TLSC_EXT.3.1

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

Section "Cryptographic support" of the [**ST**] states that the TOE presents the signature_algorithms extension in the Client Hello with the supported signature algorithms. The TOE supports SHA-384 and SHA-512 only. These are the values supported by default and cannot be changed. The TOE rejects any connection in which a certificate is not using a supported signature algorithm.

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

Section "General Common Criteria Configuration" of the [**Admin Guide**] states that the TOE supports two signature hash algorithms (SHA384, SHA512) in the signature_algorithms extension of the Client HELLO message. This is supported by default and does not need to be configured.

Testing Assurance Activities: The evaluator shall also perform the following tests:

Test 1: 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 product disconnects after receiving the server's Certificate handshake message.

Test 2: [conditional] If the client supports a DHE or ECDHE cipher suite, the evaluator shall configure the server to send a Key Exchange handshake message including a signature not supported according to the client's HashAlgorithm enumeration (for example, the server signed the Key Exchange parameters using a SHA-1 signature). The evaluator shall verify that the product disconnects after receiving the server's Key Exchange handshake message.

Test 1: The evaluator configured a server to send a certificate signed with a hash algorithm that is not supported according to the TOE's Client Hello signature_algorithms extension. The TOE properly disconnects the session after receiving the Certificate handshake message.

Test 2: The evaluator configured a server to send a Key Exchange message containing an algorithm that is not supported by the TOE according to the TOE's signature_algorithms extension. The TOE disconnects after receiving the server's Key Exchange message.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.18 TLS Client Support for Renegotiation (PKGTLS11:FCS_TLSC_EXT.4)

2.2.18.1 PKGTLS11:FCS_TLSC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance 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 cipher suite is included in the ClientHello message 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.

Test 1: The evaluator took a packet capture of a successful connection between the TOE and a TLS server. The evaluator searched for both the SCSV cipher suite and the renegotiation info field. The evaluator found that the TOE supports the SCSV ciphersuite in the Client Hello.

Test 2: The result of Test 1 shows that the Client successfully handles ServerHello messages received during the initial handshake that include the 'renegotiation_info' extension, as the Client successfully establishes a connection after checking that the ServerHello message's renegotiation_info is correctly formed. The evaluator then used a TLS server that modified the length portion of the ServerHello message to a non-zero value. The TOE correctly rejected the packet and closed the connection.

Test 3: The evaluator used a TLS server that corrupted data in the client_verify_data field. The TOE recognized the invalid value and correctly rejected the connection attempt.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.19 TLS Client Support for Supported Groups Extension (PKGTLS11:FCS_TLSC_EXT.5)

2.2.19.1 PKGTLS11:FCS_TLSC_EXT.5.1

TSS Assurance Activities: The evaluator shall verify that TSS describes the Supported Groups Extension.

Section "Cryptographic support" of the [**ST**] states that the TOE claims support for the Supported Groups Extension in the TLS handshake process. The TOE uses secp384r1 as the supported group in the TLS key exchange packets.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall also perform the following test:

Test 1: The evaluator shall configure a server to perform key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

Test 1: The evaluator connected the TOE to the ESC using each of the supported curves. The TOE connects successfully using each claimed curve.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 User data protection (FDP)

2.3.1 Encryption Of Sensitive Application Data (ASPP14:FDP_DAR_EXT.1)

2.3.1.1 ASPP14:FDP_DAR_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS. If not store any sensitive data is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory. The evaluator shall also ensure that this is consistent with the filesystem test below.

Section "User data protection" of the [**ST**] states that the TOE's sensitive data includes secret and private keys. The TOE also receives a SIP password sent by the SecuGATE. The TOE protects secret and private keys by storing the

keys using the platform provided key storage. This is in accordance with the selection made in FCS_STO_EXT.1. The TOE protects the SIP password storage on Android by saving the password in a file with the MODE_PRIVATE flag set, which is Android's default mode for creating files that give exclusive access to the application. On iOS, the TOE relies on the default iOS NSFileProtectionCompleteUntilFirstUserAuthentication data protection class to store all application files.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1. The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The TOE protects secret and private keys in accordance with FCS_STO_EXT.1. Please see ASPP14:FCS_STO_EXT.1 for the evidence that the TOE meets FCS_STO_EXT.1. The TOE protects the SIP password with mechanisms depending on the platform. The Security Target's TSS for FDP_DAR_EXT.1 states that The TOE protects the SIP password storage on Android by saving the password in a file with the MODE_PRIVATE flag set, which is Android's default mode for creating files that give exclusive access to the application. On iOS, the TOE relies on the default iOS NSFileProtectionCompleteUntilFirstUserAuthentication data protection class to store all application files.

2.3.2 Access to Platform Resources (ASPP14:FDP_DEC_EXT.1)

2.3.2.1 ASPP14:FDP_DEC_EXT.1.1

TSS Assurance Activities: None Defined

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

Section "Installing the SecuSUITE app" in the [**Admin Guide**] states that the TOE uses the following hardware resources (with justifications for each resource):

- Network connectivity - SecuSUITE requires access to a network to communicate with the SecuGATE server.
- Camera - SecuSUITE allows scanning of a QR code using the phone's camera.
- Microphone - SecuSUITE uses the microphone to record voice data.
- Bluetooth - SecuSUITE can use Bluetooth devices, such as Bluetooth headsets.
- Biometrics - The platform OS might use biometrics as the authentication factor. SecuSUITE requests access to use the platform keystore via biometrics if the phone is set up with a biometric authentication

factor.

- Notifications - SecuSUITE sends notifications to the user of incoming calls, texts and alerts.
- External Storage - SecuSUITE's text messaging system allows attachments, which are possibly stored in the phone's external storage.

The statement is consistent with the selections in the [**ST**].

Testing Assurance Activities: Platforms: Android....

The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a hardware resource is reflected in the selection.

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....

The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.

Platforms: Linux....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Platforms: Oracle Solaris....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Platforms: Apple macOS....

The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

The evaluator installed the application on each platform and determined that the TOE requests permission for

each hardware resource listed in the ST. On Android, the evaluator verified that the AndroidManifest.xml file declares all of the correct permissions. On iOS, the operating system settings as well as section 8.1 of the [**Admin Guide**] show the TOE's access permissions to hardware resources.

2.3.2.2 ASPP14:FDP_DEC_EXT.1.2

TSS Assurance Activities: None Defined

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

Section “Installing the SecuSUITE app” in the [**Admin Guide**] states that the TOE accesses contacts as a sensitive information repository.

Testing Assurance Activities: Platforms: Android....

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

Platforms: Microsoft Windows....

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

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

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

The TOE requires access to the contacts (or address book) as a sensitive information repository. The evaluator determined that the TOE requests this permission explicitly upon first use of the application.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.3 Subset Information Flow Control (VVoIPAS10:FDP_IFC.1)

2.3.3.1 VVoIPAS10:FDP_IFC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes how streaming media is not transmitted when not in a streaming media state.

Section "User data protection" of the [**ST**] states that the TOE permits SRTP traffic only when the client TOE is registered with the ESC, a call has been established with a VVoIP endpoint, the TOE is not in the 'off-hook' state, and the TOE is not in the 'mute' state.

The TOE enforces no additional information flow control policy rules, nor does it explicitly authorize or deny any information flows.

All TCP and UDP ports previously used by the TOE are closed when a call is terminated.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Testing of this component is performed through evaluation of FDP_IFF.1.

Please see section VVoIPAS10:FDP_IFF.1 for the assessment of this SFR.

2.3.4 Simple Security Attributes (VVoIPAS10:FDP_IFF.1)

2.3.4.1 VVoIPAS10:FDP_IFF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.2 VVoIPAS10:FDP_IFF.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.3 VVoIPAS10:FDP_IFF.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.4 VVoIPAS10:FDP_IFF.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.4.5 VVoIPAS10:FDP_IFF.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the TOE's enforcement of the media transmission policy and describes the conditions that are necessary for the TSF to transmit voice/video data to the operational environment.

Section "User data protection" of the [**ST**] states that the TOE enforces a media transmission policy between registered VVoIP endpoints. The VVoIP endpoints are identified by an E.164 telephone number (SIP alias). The TOE mediates the creation of SRTP channels between registered VVoIP endpoints, ensuring that both endpoints are properly identified. The TOE permits SRTP traffic only when the client TOE is registered with the ESC, a call has been established with a VVoIP endpoint, the TOE is not in the 'off-hook' state, and the TOE is not in the 'mute' state.

The TOE allows messages to be sent securely between two VVoIP endpoints. The TOE uses the same SIP-TLS protections on the secure text messages sent between two users of the TOE application.

The TOE enforces no additional information flow control policy rules, nor does it explicitly authorize or deny any information flows.

All TCP and UDP ports previously used by the TOE are closed when a call is terminated.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall set up a test environment that contains the TOE, a call control server (which may be the TOE itself), a network switch, a traffic capture tool, and a second VVoIP endpoint.

The evaluator shall then perform the following tests:

Test 1-App (conditional - software application TOE):

1. The evaluator shall execute the VVoIP application without registering it to the call control server or using P2P. The evaluator shall use the traffic capture tool to verify that the TOE does not transmit any streaming media traffic.

2. The evaluator shall place the TOE into the off-hook state (e.g., by performing a call without specifying any destination data). The evaluator shall use the traffic capture tool to verify that the TOE does not transmit any streaming media traffic.

Test 2:

1. The evaluator shall ensure the TOE is using P2P or register the TOE with the call control server and verify that the TOE is registered by checking the call control server screen with current TOE connections and by viewing the call control path traffic using the traffic capture tool.

2. The evaluator shall place the TOE into the on-hook state and verify using the traffic capture tool that no streaming media traffic is transmitted by the TOE.

3. The evaluator shall place the TOE into the off-hook state. The evaluator shall then verify that the TOE continues to not transmit streaming media traffic.

Test 3:

1. The evaluator shall ensure the TOE is using P2P or shall register the TOE with the call control server and verify that it is registered by checking the call control server with current connections and using the traffic capture tool to verify the call control path traffic.

2. The evaluator shall ensure the TOE is using P2P or shall register the second VVoIP endpoint with the call control server and verify that it has been registered by checking the call control server with current connections and using the traffic capture tool to verify the call control path traffic.

3. The evaluator shall use the TOE to dial the second VVoIP endpoint and connect a call. The evaluator shall verify that the connection is made by having a voice/video conversation with the endpoint and using the traffic capture tool to verify that a steady stream of traffic is being transmitted between the two endpoints over the media channel.

4. The evaluator shall use the TOE to put the call on mute and verify that no traffic is transmitted from the TOE over the media channel to the second VVoIP endpoint. If the TOE is registered to a call control server, the evaluator shall also verify that a mute control message is sent to the call control server and it responds.

5. The evaluator shall use the TOE to take the call off mute and verify that the streaming media traffic between the TOE and the second VVoIP endpoint is resumed.

Test 4:

1. The evaluator shall ensure the TOE is using P2P or register the TOE to the call control server and place it in the on-hook state.

2. The evaluator shall use a fuzzing tool to attempt to connect to the TOE on the full range of TCP ports used by the TSF. All ports used by the TOE should be closed except for the port that is used to communicate with the ESC.

Test 5:

1. The evaluator shall ensure the TOE and the second VVoIP endpoint are registered to a call control server or are using P2P.

2. The evaluator shall place the TOE in the on-hook state.

3. The evaluator shall use a fuzzing tool to attempt to connect to the TOE on the full range of UDP ports used by the TSF. All ports used by the TOE should be closed.

4. The evaluator shall place a call to the second VVoIP endpoint and verify the call is established. The evaluator shall capture the traffic to determine the port used by the TOE to carry the media traffic.

5. The evaluator shall hang up the call and verify that the TOE has returned to the on-hook state.

6. The evaluator shall perform fuzzing activities to verify that the port used to carry media traffic in step 4 has been closed.

Test 6 (conditional - 'The TOE is not in the hold state' is selected in FDP_IFF.1.2):

1. The evaluator shall use the TOE to put the call on hold and verify that no streaming media traffic is transmitted from the TOE over the media channel. If the TOE is registered to a call control server, the evaluator shall also verify that the VVoIP endpoint on-hold call control is sent to the call control server and it responds.

2. The evaluator shall use the TOE to take the call off hold and verify that the streaming media traffic between the TOE and the second VVoIP endpoint is resumed.

Test 1: The evaluator installed the application on each device and initialized the application. Without registering the application to the ESC, the evaluator took a packet capture of the TOE, which is in an idle (or on-hook) state. The evaluator observed the packet capture and confirmed that the TOE does not transmit any data in the idle state.

As the TOE cannot be put in an off-hook state without registering to the ESC, the evaluator used the platform's phone application to attempt to contact a number that is recognized by the TOE via the ESC contacts. The result is that the TOE does not send any media traffic in this state.

Test 2: The evaluator registered the TOE to the ESC while taking a packet capture. The evaluator observed the packet capture and verified that the registration occurred between the TOE and the ESC. The evaluator then placed the TOE in the on-hook state by opening the app while taking a packet capture. The evaluator observed the packet capture and determined that the TOE does not send any data over the media channel without making a call.

Test 3: Following Test 2, the evaluator used the TOE to call another VoIP endpoint. While the devices were ringing, the evaluator took a packet capture. The evaluator observed the traffic capture and determined that the TOE does not send any media traffic while it is in the off-hook state.

The evaluator used the TOE to call another VoIP endpoint while taking a packet capture. The evaluator picked up the call on the other end in order to establish a call connection between the TOE and the second VoIP endpoint. The packet capture shows a steady stream of media traffic between the TOE and the second VoIP endpoint.

The evaluator used the TOE to call another VoIP endpoint while taking a packet capture. The evaluator picked up the call on the other end in order to establish a call connection between the TOE and the second VoIP endpoint. The evaluator then pressed the mute button on the TOE, waited a period of time, and then unmuted the TOE. The

evaluator verified from the packet capture that the TOE stops transmitting data during the mute, and then resumes to send data after unmuting.

Test 4: The evaluator used nmap to scan all TCP ports on the TOE and found that the TOE does not open any ports unless it is instructed to communicate with the ESC.

Test 5: The evaluator used nmap to scan all UDP ports on the TOE and found that the TOE does not open any ports unless it is instructed to make a call. The evaluator then established a call and then hung up after a brief conversation. The evaluator also took a packet capture of the data exchanged between the TOE and the SIP server. The evaluator identified the UDP ports used by the TOE in the packet capture. The evaluator then performed another scan after hanging up the call and verified that all UDP ports are closed after the call ended.

Test 6: This test is not applicable as the TOE does not support the use of a hold state.

2.3.5 Network Communications (ASPP14:FDP_NET_EXT.1)

2.3.5.1 ASPP14:FDP_NET_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

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

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

Platforms: Android....

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

Test 1: The TOE does not initiate any network communication unless an action has been performed. FTP_DIT_EXT.1.1 test 1 contain packet captures for each channel that the TOE uses to communicate with the ESC. The packet captures show that each channel was initiated by a user action on the TOE.

Test 2: The evaluator tested this assurance activity under test cases 4 and 5 under VVoIPAS10:FDP_IFF.1, which show the results of a network scan of the TOE. The results are that the TOE does not open any ports until it has established a connection with the ESC.

2.4 Identification and Authentication (FIA)

2.4.1 X.509 Certificate Validation - per TD0669 (ASPP14:FIA_X509_EXT.1)

2.4.1.1 ASPP14:FIA_X509_EXT.1.1

TSS Assurance Activities: 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.

Section "Identification and authentication" of the [**ST**] states that certificates are validated as part of the authentication process when they are presented to the TOE and when they are loaded into the TOE. The TOE requires the certificate to contain a SAN extension. The following fields are verified as appropriate: SAN checks, key usages, chain validation, expiration status, and revocation status. Chain validation includes ensuring that all certificates representing a CA indicate so using the basicConstraints CA flag having a value of TRUE. Revocation checking is also performed using a Certificate Revocation List. Wildcards are not allowed in certificates.

Guidance Assurance Activities: None Defined

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

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

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,

- by omitting the basicConstraints field in one of the issuing certificates,

- by setting the basicConstraints field in an issuing certificate to have CA=False,

- by omitting the CA signing bit of the key usage field in an issuing certificate, and

- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

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

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, OCSP, OCSP Stapling, or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

The evaluator shall test revocation of the node certificate.

The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP stapling per RFC6066 is the only supported revocation method, this test is omitted.

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

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set. The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case

where the TSF is not able to establish a connection to check revocation status information, but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

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

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

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

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

Test 9: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8 with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 1: The evaluator connected to an ESC using TLS and configured to present the following invalid certificates. With each certificate, the TOE successfully identified the certificate as invalid and refused the connection attempt.

- a certificate path in which one of the issuing certificates is not a CA certificate

- missing basicConstraints field in one of the issuing certificates

- basicConstraints field in an issuing certificate to have CA=False

- CA signing bit of the key usage field in an issuing certificate is missing

- a certificate with a path length field of a valid CA field set to a value strictly less than the certificate path

Note that the "certificate path in which one of the issuing certificates is not a CA certificate" test is covered by the missing basicConstraints and CA=false test.

Test 2: The evaluator configured the ESC to use TLS. The ESC was configured with expired certificates. When the TOE attempted to connect, the connection was refused.

Test 3: The evaluator configured an ESC to use TLS. The ESC and TOE device were configured with valid certificates

allowing the TOE devices to connect to the ESC. A successful connection was made. The ESC device was then configured with a CRL revoked certificate. A revoked certificate error message was received on the TOE, and the TOE rejected the connection.

Test 4: The evaluator configured an ESC to use TLS. The ESC was configured with a certificate chain such that the subca's CRL was signed by a certificate that lacks the CRLSign purpose. The TOE correctly detects the invalid CRL signer's certificate and immediately rejects the connection attempt.

Test 5: The evaluator configured an ESC to use TLS. The ESC sends a modified certificate with a corrupted ASN.1 header. The TOE recognized that the certificate was invalid and rejected the connection attempt.

Test 6: The evaluator configured an ESC to use TLS. The ESC sends a modified certificate with a corrupted signature. The TOE recognized that the certificate was invalid and rejected the connection attempt.

Test 7: The evaluator configured an ESC to use TLS. The ESC sends a modified certificate with a corrupted public key. The TOE recognized that the certificate was invalid and rejected the connection attempt.

Test 8: The evaluator executed a control test case with a valid ECDSA certificate chain and verified that the connection was successful. The evaluator then replaced an intermediate CA with one that has explicit curves defined. The TOE successfully rejected the invalid certificate chain.

Test 9: The evaluator executed a test in which the intermediate CA in the ECDSA chain sent by the server was replaced with a certificate that has explicit format of EC parameters. The TOE immediately rejects this certificate chain and fails the connection attempt.

2.4.1.2 ASPP14:FIA_X509_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate

path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

Test 1: This has been tested in ASPP14:FIA_X509_EXT.1.1_t1 part a, where the ESC was configured with a certificate that did not contain the basicConstraints extension. An unknown CA error message was received.

Test 2: This has been tested in ASPP14:FIA_X509_EXT.1.1_t1 part b, where the ESC was configured with a certificate in which the cA flag in the basicConstraints extension is set to FALSE. An unknown CA error message was received.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.2 X.509 Certificate Authentication (ASPP14:FIA_X509_EXT.2)

2.4.2.1 ASPP14:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.2.2 ASPP14:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates. The evaluator shall examine the TSS to confirm that it describes the behavior of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the

evaluator shall ensure that the operational guidance contains instructions on how this configuration action is performed.

Section "Identification and authentication" of the [**ST**] states that the TOE automatically uses the certificates provisioned by the ESC during registration. The TOE uses certificates for TLS mutual authentication with the ESC. Certificates are checked and if found not valid, the certificates are not accepted. If an otherwise valid certificate cannot have its CRL checked to confirm that the certificate is not revoked, the certificate is not accepted.

Section "Identification and authentication" in the [**ST**] contains a description of situations that the TOE uses certificates for TLS mutual authentication with a SecuGATE SIP server. The TOE utilizes OpenSSL version 1.0.2zf to perform TLS communications and X509 certificate checking. The TOE has multiple interfaces that use TLS:

SCA Registration (TCP port 3978): This channel is for the initial set up to register the TOE to the SecuGATE.

SIP communications (TCP port 5061): This channel is used for securing the media channel setup between the TOE and SecuGATE.

Secure client authentication (TCP port 5062): This channel is used in secure communications with SecuGATE for activities such as updating configuration and secure contacts.

In all cases, the TOE checks the TLS peer's certificate. The TOE only supports mutual authentication when it connects to SecuGATE for SIP communications and secure client authentication. Certificates are provisioned by the SecuGATE SIP Server during SCA registration. The TOE uses certificates automatically, and these certificates are exchanged during TLS negotiations for the media and configuration communications with SecuGATE. If the TOE determines the certificates from the peer are not valid, the certificates are not accepted and the TOE rejects the connection attempt. If an otherwise valid certificate cannot have its CRL checked to confirm that the certificate is not revoked, the certificate is not accepted and the TOE rejects the connection.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

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

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

Test 1: The evaluator connected the TOE to the ESC, and the evaluator had a valid revocation server set up. The TOE is able to download the CRLs and verify the revocation status of the peer's certificate, and the connection

succeeds. The evaluator then connected the TOE to the ESC with a test setup in which the TOE was unable to reach the revocation server. The TOE attempts to connect to the unreachable CRL server, but the connection fails, so the TOE rejects the connection attempt to the ESC.

Test 2: The TOE checks the peer's certificate validity before it checks the certificate's revocation status. Thus, the TOE is able to detect and reject an invalid certificate before moving on to check its revocation.

2.5 SECURITY MANAGEMENT (FMT)

2.5.1 SECURE BY DEFAULT CONFIGURATION (ASPP14:FMT_CFG_EXT.1)

2.5.1.1 ASPP14:FMT_CFG_EXT.1.1

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

Section "Security management" in the [**ST**] states that the TOE does not install with any default credentials. The TOE also does not require any type of credential to provide access to the TOE.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If the application uses any default credentials the evaluator shall run the following tests.

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

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

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

Test 1: This test is not applicable as the TOE does not contain default credentials.

Test 2: This test is not applicable as the TOE does not contain default credentials.

Test 3: This test is not applicable as the TOE does not contain default credentials.

2.5.1.2 ASPP14:FMT_CFG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

For Android, the evaluator determined that there are no world writable files in the TOE's data directory by executing the "find -L . -perm /002" command inside the application's data directories. The command did not return any files. The TOE also does not store any data in external storage.

For iOS, the evaluator searched the decompiled application and determined that the TOE uses the default

protectioncompleteuntilfirstuserauthentication file protection employed by iOS.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Supported Configuration Mechanism - per TD0624 (ASPP14:FMT_MEC_EXT.1)

2.5.2.1 ASPP14:FMT_MEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section "Security management" of the [**ST**] states that the TOE stores application configuration options for Android platforms in an XML file at location /data/data/package/shared_prefs. For iOS platforms the TOE stores application configuration options using the user defaults system. More specifically, the TOE uses the UserDefaults and NSUserDefaults APIs. The TOE's SIP configuration is stored locally, but established and modified by the remote SCA or SIP server. Neither the TOE nor platform offer the ability to configure the TOE's SIP settings locally.

Component Guidance Assurance Activities: None Defined

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

Platforms: Android....

The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least one file exists at location /data/data/package/shared_prefs/ (for SharedPreferences) and/or /data/data/package/files/datastore (for DataStore), where the package is the Java package of the application. For SharedPreferences the evaluator shall examine the XML file to make sure it reflects the changes made to the configuration to verify that the application used SharedPreferences and/or PreferenceActivity to store the configuration data. For DataStore the evaluator shall use a protocol buffer analyzer to examine the file to make sure it reflects the changes made to the configuration to verify that the application used DataStore to store the configuration data.

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

If ' implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption' is selected, for all configuration options listed in the TSS as being stored and protected using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

For Android, the evaluator searched the platform's filesystem and verified that when a TOE setting was changed, a file in the /data/data/<package>/shared_prefs directory was updated.

For iOS, the evaluator searched the decompiled application and verified that the TOE uses the default iOS UserDefaults and NSUserDefaults APIs to store configuration data.

2.5.3 Specification of Management Functions (ASPP14:FMT_SMF.1)

2.5.3.1 ASPP14:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

The TOE allows specifying the SIP server as a management function. Section "Activating the SecuSUITE app" of the [**Admin Guide**] states the procedures for specifying the server URL for connecting to the SIP server. The operator must either scan the QR code provided by SecuGATE or type in the server and activation code manually.

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

The evaluator exercised the selected management functions in the TOE. The evaluator was able to specify the server after entering in the activation code.

2.5.4 Specification of Management Functions (VVoIP Communications) (VVoIPAS10:FMT_SMF.1/VVoIP)

2.5.4.1 VVoIPAS10:FMT_SMF.1.1/VVoIP

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS provides a description of the TOE initial configuration and describes the ability of the TSF to manage the functions that are defined in the SFR, including how each function is managed (e.g. manually configured, applied via downloaded configuration file).

Section "Security management" of the [**ST**] states that the TOE's only management function is providing the ability to register the TOE to an ESC manually. The TOE contains no default configuration after installation. The TOE requires the operator to enter in the server address to register the TOE with the ESC. This is done either by manual input or by scanning a QR code.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions on configuring any functionality specifically related to VVoIP.

The TOE allows registration of the TOE to the ESC as a management function. Section “Activating the SecuSUITE app” of the [**Admin Guide**] provides the procedures for registering with the ESC.

Component Testing Assurance Activities: The evaluator shall perform the following tests, depending on the selections made:

Test 1 (conditional - 'register the TOE to an ESC [manually]' is selected):

1. On the TOE, input IP address, gateway address, and subnet mask.

2. If the operational environment is deployed in a manner such that the configuration server and ESC are two distinct servers, input the addresses for each; otherwise, input the ESC address.

3. Save configuration.

4. Verify the TOE registers to the ESC.

Test 2 (conditional - 'register the TOE to an ESC [via DHCP server]' is selected):

1. On the TOE, input the DHCP server address.

2. Save configuration.

3. Verify by traffic capture that the TOE receives all needed IP addresses.

4. Verify by examining the IP address on the TOE.

5. Verify the TOE registers to the ESC.

Test EAs to verify that the TOE can act as a VVoIP call control server when using P2P (if it is selected) are performed as part of testing for FDP_IFF.1/CallControl.

Test EAs to verify the configuration of audit behavior are performed as part of testing for FAU_STG_EXT.1.

Test EAs to verify the modification of transmission of audit data to an external IT entity are performed as part of testing for FAU_STG_EXT.1.

Test EAs to verify that the idle call termination period can be specified are performed as part of testing for FTA_SSL.3/Media, specifically Test 2 and Test 3.

Test EAs to verify that the vocoder can be specified are performed as part of testing for FCO_VOC_EXT.1.

Test 1: The evaluator demonstrated that the TOE registers to the ESC manually. The evaluator input the server URL in the TOE's initial screen. The URL can be either an IP address or a hostname (which resolves to an IP address). The TOE also uses this IP address as the gateway. The subnet mask of 255.255.255.255 is automatically applied as the IP address is a specific address. After submitting an additional registration code, the TOE is successfully registered to the ESC.

2.6 Privacy (FPR)

2.6.1 User Consent for Transmission of Personally Identifiable (ASPP14:FPR_ANO_EXT.1)

2.6.1.1 ASPP14:FPR_ANO_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section "Privacy" of the [**ST**] states that the TOE does not collect or transmit any PII.

Component Guidance Assurance Activities: None Defined

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

This test is not applicable as the TOE does not collect or transmit any PII.

2.7 Protection of the TSF (FPT)

2.7.1 Anti-Exploitation Capabilities (ASPP14:FPT_AEX_EXT.1)

2.7.1.1 ASPP14:FPT_AEX_EXT.1.1

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

Section "Protection of the TSF" of the [**ST**] states that the TOE is compiled to enable ASLR.

Android's application management requires application updates to be signed with an Android key, thus allowing the secure updates of its applications. The Android OS Linux kernel is capable of ASLR (address space layout randomization), ensuring that no application uses the same address layout on two different devices.

The TOE libraries are also compiled with the '-fstack-protector-all -fno-exceptions' flags in order to enable ASLR and stack-based buffer over flow protections. On iOS the ASLR feature (-pie) is not set by a compiler flag, because it is on by default on the C-language compiler. This setting is required by the Apple App store.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.

Platforms: Android....

The evaluator shall run the same application on two different Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool. Connect via ADB and inspect /proc/PID/maps. Ensure the two different instances share no memory mappings made by the application at the same location.

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

For Android, the evaluator took two process ID maps from an instance of the TOE on two separate Android debug devices. The evaluator compared the two and determined that the TOE does not share any memory mapping locations.

For iOS, the evaluator searched for calls to mmap and found that existing calls to mmap come from third party libraries statically linked in the application. There is no evidence that these mmap calls request a static position in memory.

2.7.1.2 ASPP14:FPT_AEX_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

o mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked.

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

o mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Oracle Solaris....

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

o mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and

o mprotect is never invoked with the PROT_EXEC permission.

Platforms: Apple macOS....

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

The evaluator used decompiled versions of the application on both platforms to search for the mmap and mprotect.

For Android, the evaluator verified that the TOE does not call mprotect. There are a few classes in Android that call upon mmap, but the classes are not security relevant. These classes handle font types, font weight, italics style, and emojis. The calls to mmap here do not invoke either the PROT_EXEC or PROT_WRITE permissions.

For iOS, the evaluator verified that the TOE does not use mprotect.

2.7.1.3 ASPP14:FPT_AEX_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threatprotection/windows-defender-exploit-guard/customize-exploit-protection.

If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

According to the AA, applications on both Android and iOS cannot disable security features, thus no testing is required.

2.7.1.4 ASPP14:FPT_AEX_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

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

Platforms: Linux....

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

Platforms: Oracle Solaris....

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

Platforms: Apple macOS....

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

For Android, the evaluator captured a listing of the files before and after normal usage of the application. The evaluator executed the TOE and used it to register to the ESC server as well as to make a call. Comparison of the two output files shows that the TOE application does write data to the executable directory.

For iOS, the TOE meets this requirement since the platform forces applications to write all data within the application working directory (sandbox).

2.7.1.5 ASPP14:FPT_AEX_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

Platforms: Microsoft Windows....

Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

For PE , the evaluator will disassemble each and ensure the following sequence appears:

```
mov rcx, QWORD PTR [rsp+(...)]

xor rcx, (...)

call (...)
```

For ELF executables, the evaluator will ensure that each contains references to the symbol _stack_chk_fail.

Tools such as Canary Detector may help automate these activities.

For both iOS and Android, the evaluator used decompiled versions of each application to search for instances of the stack_chk_guard and stack_chk_fail keywords. The evaluator found that on Android, the TOE contains the stack_chk_fail symbol. On iOS, the TOE contains both the stack_chk_fail and stack_chk_guard symbols. Thus, stack

protections exist in both applications.

Component TSS Assurance Activities: None Defined **Component Guidance Assurance Activities**: None Defined **Component Testing Assurance Activities**: None Defined

2.7.2 Use of Supported Services and APIs (ASPP14:FPT_API_EXT.1)

2.7.2.1 ASPP14:FPT_API_EXT.1.1

TSS Assurance Activities: None Defined **Guidance Assurance Activities**: None Defined **Testing Assurance Activities**: None Defined
Component TSS Assurance Activities: The evaluator shall verify that the TSS lists the platform APIs used in the application.

In the [**ST**], section "Android platform interfaces invoked by the TOE" lists the Android APIs used by the TOE. Section "iOS platform interfaces invoked by the TOE" lists the iOS APIs used by the TOE.

Component Guidance Assurance Activities: None Defined **Component Testing Assurance Activities**: The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

The evaluator compared the list of the TOE's claimed APIs with the list of supported APIs on each respective platform and ensured that all APIs listed in the TSS are supported.

2.7.3 Software Identification and Versions (ASPP14:FPT_IDV_EXT.1)

2.7.3.1 ASPP14:FPT_IDV_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'other version information' is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology.

Section "Protection of the TSF" of the [**ST**] states that the TOE provides a multi-part unique release number. The version includes a major release, minor release, build number, date, and whether the build is a debug or release.

Component Guidance Assurance Activities: None Defined

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

The evaluator installed the TOE application and verified that the TOE's versioning information can be found either in the application's settings or in the platform's settings. This is true for both Android and iOS platforms.

2.7.4 Use of Third Party Libraries (ASPP14:FPT_LIB_EXT.1)

2.7.4.1 ASPP14:FPT_LIB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

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

The TOE compiles static libraries into its binary for each platform. The TOE does not have dynamic libraries. Static libraries that are built into the executable do not have header information. In order to verify that the libraries are in the executable, the evaluator decompiled each of the application packages (.apk for Android and .ipa for iOS).

The evaluator then searched for the existence of each library by using each library name as a search term. The evaluator found that the claimed libraries are included in each respective application package.

2.7.5 Integrity for Installation and Update (ASPP14:FPT_TUD_EXT.1)

2.7.5.1 ASPP14:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall check to ensure the guidance includes a description of how updates are performed.

The TOE relies on platform app stores to securely update the TOE. Section "Client update via App Stores" in the [**Admin Guide**] includes procedures to check for an update in the app store that corresponds to each platform.

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

The evaluator followed operational guidance to check for an update in each app store. For the TOE on both platforms, the update button did not exist, which meant that the current version was the most recent and that no update was needed.

2.7.5.2 ASPP14:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall verify guidance includes a description of how to query the current version of the application.

Section “Client Software Version” of the [**Admin Guide**] provides the procedures for querying the current version of the application. The version can be viewed either through the application (through Settings > About SecuSUITE), or viewed by each platform settings page.

Testing Assurance Activities: The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

For both iOS and Android, the evaluator was able to view the TOE's current version through the application's settings page as well as in each OS platform's Settings menu. The evaluator verified that the current major version (5.0) matches that of the documented and installed version.

2.7.5.3 ASPP14:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall verify that the application's executable files are not changed by the application.

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

For all other platforms, the evaluator shall perform the following test:

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

On Android, the evaluator used a debug Android device to record the hash of every executable in the TOE's installation directory. The evaluator then proceeded to use the TOE normally. Afterwards, the evaluator queried the hash for each executable again. The evaluator confirmed that the hashes remained the same.

On iOS, this requirement is met because the platform forces applications to write all data within the application working directory (sandbox).

2.7.5.4 ASPP14:FPT_TUD_EXT.1.4

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

Section "Protection of the TSF" of the [**ST**] states that TOE updates are signed with the Blackberry software signing key associated with each platform. Blackberry is an authorized source as they are a vendor that obtained official signing keys from Google and Apple in order to publish the TOE application on both the Google Play Store and Apple App Store. Downloaded TOE updates (apps) are verified using digital signatures in accordance with each platform. Both app stores require developers to have a key for each respective app store. Blackberry digitally signs the application with each key on each respective platform in accordance with each supported mobile device Security Targets.

Candidate updates are obtained via each TOE platform's respective app store or via an MDM. The TOE platform always checks the signatures of the update files before the updates are applied, whether the app is obtained via the app store or through an MDM.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.5.5 ASPP14:FPT_TUD_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS identifies how the application is distributed.

Section "Protection of the TSF" of the [**ST**] states that the application is distributed as an Android application package (APK) on an Android platform and using the IPA format on an iOS platform. The TOE is distributed as an additional software package, and is installed using the respective app store of each platform.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If 'with the platform' is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If 'as an additional package' is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

This test has been performed in ASPP14:FPT_TUD_EXT.2.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7.6 Trusted Update (VVoIPAS10:FPT_TUD_EXT.1)

2.7.6.1 VVoIPAS10:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: There is no change to the EAs specified for this SFR in the App PP. Note however that the following additional configuration steps are necessary in order for this testing to be performed:

- The evaluator shall deploy a call control server or dedicated file server in the TOE's operational environment

- The evaluator shall load valid and invalid candidate updates to the call control server or dedicated file server

- The evaluator shall configure the TOE to use the call control server or dedicated file server as its source for software/firmware updates

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

The evaluator configured an Airwatch MDM/MAS deployment accessible by the TOE to manage the TOE. The evaluator configured valid and invalid TOE software updates in the MAS Server. The evaluator then enrolled the TOE platform in the MDM and used the associated MAS to deploy updates to the TOE. Note that Airwatch does not implement any of the required security functions; the applicable functions subject to testing are implemented in the TOE and its host platform.

2.7.7 Integrity for Installation and Update – per TD0628 and TD0664 (ASPP14:FPT_TUD_EXT.2)

2.7.7.1 ASPP14:FPT_TUD_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: If a container image is claimed the evaluator shall verify that application updates are distributed as container images.

If the format of the platform-supported package manager is claimed, the evaluator shall verify that application updates are distributed in the correct format. This varies per platform:

Platforms: Android....

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

Platforms: Microsoft Windows....

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

Platforms: Apple iOS....

The evaluator shall ensure that the application is packaged in the IPA format.

Platforms: Linux....

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

Platforms: Oracle Solaris....

The evaluator shall ensure that the application is packaged in the PKG format.

Platforms: Apple macOS....

The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format.

The evaluator successfully verified that the application is packaged in the standard Android application package (APK) on Android platforms, and the application is packaged as an IPA (iOS App Store Package) on an Apple iOS platform.

2.7.7.2 ASPP14:FPT_TUD_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Platforms: Android....

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

Platforms: Apple iOS....

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

All Other Platforms...

The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

According to the AA, the TOE meets this requirement as both the Android and iOS applications are forced to write all files in the application's working directory in each platform's respective directory structure.

2.7.7.3 ASPP14:FPT_TUD_EXT.2.3

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

Section "Protection of the TSF" of the [**ST**] states that TOE updates are signed with the Blackberry software signing key associated with each platform. Blackberry is an authorized source as they are a vendor that obtained official signing keys from Google and Apple in order to publish the TOE application on both the Google Play Store and Apple App Store. Downloaded TOE updates (apps) are verified using digital signatures in accordance with each platform's app store. Both app stores require developers to have a key for each respective app store. Blackberry digitally signs the application with each key on each respective platform in accordance with each supported mobile device Security Targets.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8 TOE ACCESS (FTA)

2.8.1 /MEDIA TSF-INITIATED TERMINATION (MEDIA CHANNEL) (VVOIPAS10:FTA_SSL.3/MEDIA)

2.8.1.1 VVOIPAS10:FTA_SSL.3.1/MEDIA

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS specifies whether idle calls are terminated by default after a certain amount of time or by an administrator-configurable interval.

Section "TOE access" of the [**ST**] states that only SecuGATE can configure the SRTP timeout value. SecuGATE enforces a default timeout value of 15 seconds. SecuGATE configures this timeout value on the TOE via the SCA registration.

Component Guidance Assurance Activities: If the idle call timeout period is administrator-configurable, the evaluator shall verify that the operational guidance includes instructions for how to configure this, as well as what the minimum and maximum allowed values are.

The SRTP timeout value is not configurable on the TOE. The default is 15 seconds.

Component Testing Assurance Activities: The evaluator shall set up a test environment that contains the TOE, a call control server (which may be the TOE itself), a configuration server (if used to communicate configuration changes to idle timeout period), a network switch, a traffic capture tool, and a second VVoIP endpoint.

The evaluator shall then perform the following tests:

Test 1:

1. Deploy the TOE in a default configuration (i.e. without any administrative override applied to the idle timeout value).

2. Ensure the TOE is using P2P or register the TOE with the call control server and verify that it is registered by viewing its status on the ESC and capturing the call control path traffic.

3. Ensure the second VVoIP endpoint is using P2P or register the second VVoIP endpoint with the call control server and verify that it is registered by viewing its status on the call control server and capturing the call control path traffic.

4. Use the TOE to dial the second VVoIP endpoint and establish a call. Verify the call was established by holding a conversation between the two peers and capturing the streaming media traffic that is transmitted between them.

5. Power down the second VVoIP endpoint while the call is active. Observe that the TOE stops transmitting media after the default period of time specified in the ST.

Test 2 (conditional - 'an administrator-configurable interval' is selected in FTA_SSL.3.1/Media):

1. Deploy the TOE in a default configuration (i.e. without any administrative override applied to the idle timeout value).

2. Ensure the TOE is using P2P or register the TOE with the call control server and verify that it is registered by viewing its status on the call control server and capturing the call control path traffic.

3. Configure the TOE's idle timeout period for the shortest period of time that is supported.

4. Ensure the second VVoIP endpoint is using P2P or register the second VVoIP endpoint with the call control server and verify that it is registered by viewing its status on the call control server and capturing the call control path traffic.

5. Use the TOE to dial the second VVoIP endpoint and establish a call. Verify the call was established by holding a conversation between the two peers and capturing the streaming media traffic that is transmitted between them.

6. Power down the second VVoIP endpoint while the call is active. Observe that the TOE stops transmitting media after the period of time configured in Step 3.

Test 3 (conditional - 'an administrator-configurable interval' is selected in FTA_SSL.3.1/Media):

Repeat Test 2 but in Step 3, configure the idle timeout value to be the longest period of time that is supported as opposed to the shortest.

Test 1: The evaluator registered the TOE application to the ESC without changing the default SRTP timeout value. The evaluator then placed a call from the TOE application to a second VoIP endpoint. The evaluator then powered down the second VoIP endpoint and observed that the TOE stops transmitting media traffic after the default timeout value. This test was performed for both Android and iOS platforms.

Test 2: Not applicable as the timeout value is not administrator configurable in the TOE.

Test 3: Not applicable as the timeout value is not administrator configurable in the TOE.

2.9 Trusted path/channels (FTP)

2.9.1 Protection of Data in Transit - per TD0655 (ASPP14:FTP_DIT_EXT.1)

2.9.1.1 ASPP14:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The TOE does not leverage platform-provided functionality for protection of data in transit. Section "Trusted path/channels" of the [**ST**] states that the TOE implements functionality to encrypt all communication using TLS, SRTP or SIP over TLS. The TOE uses these protocols to protect SIP signaling, VoIP calls, and secure text messaging.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests.

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

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

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

Platforms: Android....

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

Platforms: Apple iOS....

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

Test 1: The evaluator took a packet capture of a full connection of the TOE to the ESC for registration and placing calls. The evaluator used both the Android and iOS devices to establish a call to capture media traffic between the two devices. The evaluator confirmed that each channel (ESC communication and phone calls) is encrypted accordingly. The text messaging system protections tests are covered by the testing of the SIP-TLS channels for VoIP calls as the same SIP-TLS channel is used.

Test 2: The packet capture analysis for each channel in test 1 shows that there is no plaintext data transmitted.

Test 3: The ESC assigns a phone number to identify the TOE. The TOE also uses this information as a SIP username. The SIP client also calculates a SIP password based on a nonce provided by the ESC. The evaluator opened up the encrypted call packet capture in a hex editor and then search for the SIP username and a part of the SIP password. The result is that these credentials are adequately protected via TLS as the evaluator cannot find the values in the encrypted TLS channels.

For iOS, the evaluator looked through a decompiled application and confirmed that the applications Info.plist file does not contain NSAllowsArbitraryLoads or NSExceptionAllowsInsecureHTTPLoads keys.

2.9.2 Protection of Data in Transit (VVoIPAS10:FTP_DIT_EXT.1)

2.9.2.1 VVoIPAS10:FTP_DIT_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.9.3 Inter-TSF Trusted Channel (Signaling Channel) (VVoIPAS10:FTP_ITC.1/Control)

2.9.3.1 VVoIPAS10:FTP_ITC.1.1/Control

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the ability of the TOE to use SIP and/or H.323 with TLS.

Section "Trusted path/channels" of the [**ST**] states that the TOE uses SIP for all data transmitted between itself and

an Enterprise Session Controller for the purpose of communicating with another VVoIP endpoint. The SIP communication occurs within the context of an already established TLS session. The TOE or ESC can initiate SIP communication in the existing TLS session.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1/Control requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

Test 1: The evaluators shall ensure that communications using each protocol with a call control server is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for a connection to a call control server using each claimed protocol, physically interrupt the connection to the call control server for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the call control server. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further EAs are associated with the specific protocols.

Test 1: As shown in the test results for ASPP14:FTP_DIT_EXT.1, the evaluator confirmed that all communications from the TOE to the ESC and to another VoIP endpoint is successful. The communications are encrypted accordingly.

Test 2: The test results for ASPP14:FTP_DIT_EXT.1 show that the communications are TOE initiated.

Test 3: The test results for ASPP14:FTP_DIT_EXT.1 show that the data is encrypted and not transmitted in plaintext.

Test 4: In order to test physical disruption of the TOE's SIP traffic, the evaluator took a packet capture of a call attempt from the ESC and focused on tcp port 5061, which is the port used for SIP signaling traffic. The evaluator registered both the Android and iOS devices as well as another VoIP endpoint to the ESC. The evaluator then made a call from the TOE (on both the Android and iOS platforms) to the third VoIP endpoint but left the call ringing. The TOE is continually sending SIP traffic through the TLS tunnel by letting the application continue to ring. The evaluator took a packet capture of the call attempt.

For the MAC layer disruption, the evaluator initiated a call from the Android device to iOS and let the phones ring. The evaluator disconnected the connection between the phones and the ESC. The evaluator waited a short time and then reconnected. After restoring connectivity, with the ESC, the phones resumed protected SIP data transfer using TLS.

For the App layer disruption, the evaluator repeated the above test, only this time the evaluator waited a longer period of time. This causes the TOE application to close the call attempt and close out the SIP signaling, thus requiring the operator to set up a new call. The TOE set up a new TLS session for the new call session. This shows that the TOE is still able to protect traffic after an app layer timeout.

2.9.4 Inter-TSF Trusted Channel (Media Channel) (VVoIPAS10:FTP_ITC.1/Media)

2.9.4.1 VVoIPAS10:FTP_ITC.1.1/Media

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the trusted channel will use SRTP or H.323/H.235 in accordance with the selections made in the SFR.

Section "Trusted path/channels" of the [**ST**] states that the TOE uses SRTP as a trusted channel to communicate with a proxy on SecuGATE. Using this proxy, the TOE can communicate with another VVoIP endpoint for media traffic. The STRP cryptographic parameters are passed between the TOE and the other VVoIP endpoint using SIP messages protected by TLS (traffic that is part of FTP_ITC.1/Control). The TOE can either initiate or receive voice/video media traffic.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The vendor shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1/Media requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

Test 1: The evaluators shall ensure that communications using each protocol with another VVoIP endpoint is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for a connection to another VVoIP endpoint device using each claimed protocol, physically interrupt the connection to that remote endpoint for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the MAC layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the remote endpoint. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Test 1: As shown in the test results for ASPP14:FTP_DIT_EXT.1, the evaluator confirmed that all communications from the TOE to the ESC and to another VoIP endpoint is successful. The communications are encrypted accordingly.

Test 2: The test results for ASPP14:FTP_DIT_EXT.1 show that the communications are TOE initiated.

Test 3: The test results for ASPP14:FTP_DIT_EXT.1 show that the data is encrypted and not transmitted in plaintext.

Test 4: The evaluator first registered both the Android and the iOS devices to the ESC as well as a third VoIP endpoint. The evaluator set up valid calls between the TOE on Android and iOS with the third VoIP endpoint. The evaluator physically disrupted the network such that all devices no longer had access to the ESC.

For the MAC layer disruption, the evaluator established a call between the two devices. The evaluator then disconnected the connection between the devices and the ESC and waited a short amount of time such that the MAC layer will time out. The evaluator re-established the link between the devices and the ESC. The evaluator observed the media channel is still protected.

For the App layer disruption, the evaluator repeated the above test with the exception that the wait time was much longer (about 15 minutes). This caused the devices to end the calls on the devices and close out the TLS connection, and the operator is forced to establish a new call (with a new TLS session). After re-establishing the link between the devices and the ESC, the evaluator showed that the TOE needed to create a new TLS session to establish another call.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: 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 evaluation activities described in Section 5.1, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other assurance activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: Some of the contents of the operational guidance will be verified by the assurance activities in Section 5.1 and evaluation of the TOE according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE. The documentation must describe the process for verifying updates to the TOE by verifying a digital signature - this may be done by the TOE or the underlying platform. The evaluator shall verify that this process includes the following steps: Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory). Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

Section “General Common Criteria Configuration” in the [**Admin Guide**] states that the TOE uses an internal cryptographic module which uses NIST approved algorithms. The cryptographic engine cannot be changed by the security administrator or the SecuSUITE user.

Section “Client update via App Stores” in the [**Admin Guide**] provides the procedures for updating the client using each platform’s app store. This means that the TOE’s application data is verified by the platform. The procedures include checking for a possible update via the “Updates pending” section. If this section is present, then the

operator will select the Update button to update the application. Once the update finishes, the operator must then check the versions to ensure that the update is successful.

Section “Security Functionality covered by the Common Criteria Evaluation” in the [**Admin Guide**] contains a paragraph that explains the security functionality covered by the evaluation. The section claims that the security functions tested are outlined in the [ST], and these functions include TLS trusted channels, x.509 authentication, certificate validation and signature checking. The section also explains an additional layer of SRTP session key protection that is not covered by the evaluation. The section mentions that Breakout and Secure Landing are not covered by this evaluation.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: 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.

Section “Supported devices and firmware” in the [**Admin Guide**] states the claimed platforms, which are consistent with the claims made in the Security Target.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: 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.

Section "TOE Description" of the [**ST**] contains the TOE identifier. The TOE is comprised of the SecuSUITE v5.0 and SteelBox v5.0. Please note that the SteelBox Client is a branded version of the SecuSUITE client that is identical from functional and security implementation perspective. The SteelBox client is distributed by BlackBerry’s partner CACI and differs basically in the used UI assets and product publishing.

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 [**Admin Guide**] adequately shows that the TOE's version 5.0 is consistent with the 5.0 version listed in the [**ST**]. The [**Admin Guide**] references the TOE's version in sections "Variants of the TOE" and "Supported devices and firmware".

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: The 'evaluation evidence required by the SARs' in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

The evaluator verified that the [**ST**], TOE and [**Admin Guide**] are all labeled with identifiable versions. The evaluator checked the TOE version during testing by examining the actual devices used for testing.

The evaluator shall ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

The [**Admin Guide**] identifies a specific version of the TOE for which it is relevant and identifies the platforms for the evaluation.

3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activities: 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 entire application.

The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described. 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. The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE.

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.

Section "Protection of the TSF" of the [**ST**] states that the vendor provides timely security updates for the TOE in case vulnerabilities have been discovered. Reported vulnerabilities and defects are investigated and rated based on the threat and result of the impact analysis and then scheduled for an upcoming bug fix release based on the severity. BlackBerry aims for a security update between 10 and a maximum of 50 days. Third party library updates are also included as a part of the TOE's update. BlackBerry accepts vulnerability reports through the BlackBerry form at <https://www.blackberry.com/us/en/forms/enterprise/contact-us>.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's 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, 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 evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results.

The following diagram depicts the evaluator's test environment:

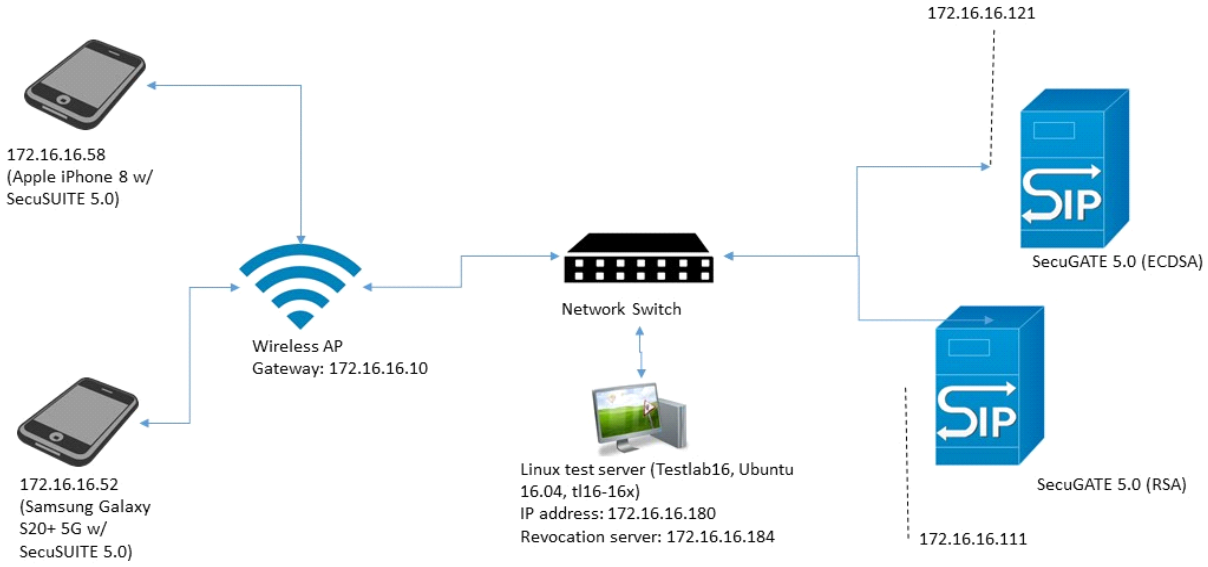

Figure 1 Test Setup

TOE Platforms:

- SecuSUITE 5.0 on Samsung Galaxy S20+ 5G running Android 11
- SecuSUITE 5.0 on Apple iPhone 11 running iOS 14

Supporting Products:

- SecuGATE 5.0
- SecuSUITE 4.0/5.0 on Samsung Galaxy A71 running Android 11 (user debug build)
- SecuSUITE 4.0/5.0 on Samsung Galaxy A52 running Android 11 (user debug build)

Supporting Software:

- SSH Client - OpenSSH version 7.2p2
- SSH Client - Putty version 0.68
- Wireshark version 4.0.1

- Nmap version 7.01
- Stunnel 5.30 with OpenSSL 1.0.2g
- Stunnel 5.65 with OpenSSL 1.1.1q
- Apple oTool for OSX Mojave (10.14)
- Apple iTunes 12
- Android Debug Bridge 1.0.41
- HxD version 2.3.0

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses.

The evaluator searched the National Vulnerability Database (<https://web.nvd.nist.gov/vuln/search>), Vulnerability Notes Database (<http://www.kb.cert.org/vuls/>). The following search terms were used: "webrtc ", "Boost ", "ZLIB ", "BZip2 ", "Openssl 1.0.2", "pjproject ", "PocoCpp ", "libphonenumber ", "icu 63.1", "protobuf ", "opus ", "silk ", "libsrtp ", "secusmart", "secusuite", "pcsclite ", "android.tools", "androidx.lifecycle ", "androidannotations ", "bumptech.glide ", "PhotoView", "androidx.appcompat ", "swiperefreshlayout", "androidx.biometric", "androidx.constraintlayout ", "androidx.recyclerview ", "android.material ", "joda-time", "spongycastle", "facebook.shimmer ", "androidx.emoji", "android.gms", "firebase ", "zxing", "YYImage ", "cryptocomply", "blackberry", "caci steelbox". An initial search was conducted on 11/14/2022.

The evaluator searched the vulnerability sites above using the keywords shown above. At that time all issues found were considered and a resolution determined for each issue. The results of that analysis are shown below. Updates to this public search were also performed. During an update the evaluator searched each site, for each term, but considered only new or changed issues that were returned by the public sites which appeared since the previous search. For the update, every issue was considered and a resolution was determined for each issue. The evaluator determined that there are no open public issues.

The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious. The evaluator documents the sources consulted and the vulnerabilities found in the report. For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

This test is not applicable as the TOE is built for Android and iOS only.