The RSASP1 Signature Primitive Validation System (RSASP1VS)

January 8, 2013

Sharon S. Keller

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

Update Log

1 Introduction

This document, *RSASP1 Signature Primitive Validation System (RSASP1VS),* specifies the procedures involved in validating implementations of the RSASP1 signature primitive specified in Section 5.2 of the *PKCS#1 v2.1: RSA Cryptography Standard (June 14, 2002)* [1]. This primitive is used by both the RSA SSA-PKCS1-v1_5 and the RSA SSA-PSS signature schemes referenced in FIPS186-3, *Digital Signature Standard (DSS)* [2] and specified in [1]. The validation testing of the RSASP1 primitive validates the correctness of the exponentiation function portion of the signature generation function by both the RSA SSA-PKCS1-v1_5 and RSA SSA-PSS signature schemes.

The RSASP1VS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the RSASP1VS. It defines the purpose, the design philosophy, and the high-level description of the validation process for the RSASP1 signature primitive. The requirements and procedures to be followed by those seeking formal validation of an implementation of the RSASP1 primitive are presented. The requirements described include the specification of the data communicated between the IUT and the RSASP1VS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the RSASP1VS.

A set of RSASP1 primitive test vectors for both the RSA SSA-PKCS1-v1_5 and the RSA SSA-PSS signature schemes is available on the http://csrc.nist.gov/groups/STM/cavp/index.html website for testing purposes.

2 Scope

This document specifies the validation tests required to validate IUTs for conformance to the RSASP1 primitive utilized by both the RSA SSA-PKCS1-v1_5 and the RSA SSA-PSS signature schemes. The RSASP1 primitive performs the exponentiation function resulting in the signature. In applications using PIV cards, this exponentiation function is performed on the card. The RSASP1VS provides testing to determine the correctness of the implementation of the primitive. The RSASP1VS consists of a different validation test for each signature scheme. It determines if the RSASP1 implementation produces the expected signature given a set of encoded message inputs.

3 Conformance

The successful completion of the individual validation test(s) contained within the RSASP1VS that pertains to the RSA signature scheme(s) being validated is required to claim conformance to the RSASP1 primitive. Testing for the cryptographic module in which the application-specific algorithm is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules* [3].

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	MEANING
CST laboratory	Cryptographic Security Testing laboratory that operates the RSASP1VS
IUT	Implementation Under Test
PKCS	Public Key Cryptography Standards

4.2 Abbreviations

ABBREVIATION	MEANING
ANS	American National Standard
RSASP1VS	RSASP1 Signature Primitive Validation System
FIPS	Federal Information Processing Standard
IUT	Implementation Under Test

5 Design Philosophy of the RSASP1 Primitive Validation System

The RSASP1VS is designed to test conformance to specifications in Section 5.2 of PKCS#1 v2.1 rather than provide a measure of a product's security. The validation tests are designed to assist in the detection of accidental implementation errors, and are not designed to detect intentional attempts to misrepresent conformance. Thus, validation should not be interpreted as an evaluation or endorsement of overall product security.

The RSASP1VS has the following design philosophy:

1. The RSASP1VS is designed to allow the testing of an IUT at locations remote to the RSASP1VS. The RSASP1VS and the IUT communicate data via *REQUEST (.req)* and *RESPONSE (.rsp)* files. The RSASP1VS also generates *SAMPLE (.sam)* files to provide the IUT with an example of the *RESPONSE* file format.

2. The testing performed within the RSASP1VS uses statistical sampling (i.e., only a small number of the possible cases are tested); hence, the successful validation of a device does not imply 100% conformance with the Recommendation.

6 RSASP1VS Tests

When applied to an IUT, the RSASP1VS provides testing to determine the correctness of the RSASP1 function. Each IUT supported RSA signature scheme is tested individually. The RSASP1VS consists of a single validation test for each supported RSA signature scheme. The RSASP1VS requires the vendor to specify the RSA signature scheme to be tested (e.g., RSA_SSA-PKCS1-v1_5, RSA_SSA-PSS). Separate files will be generated for each RSA signature scheme tested.

6.1 Configuration Information

To initiate the validation process of the RSASP1VS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of the RSASP1 Signature component. The vendor's implementation is referred to as the Implementation Under Test (IUT). The request for validation includes background information describing the IUT along with information needed by the RSASP1VS to perform the specific tests. More specifically, the request for validation includes:

1. Cryptographic algorithm implementation information

   a) Vendor Name

   b) Implementation Name;

   c) Implementation Version;

   d) Indication if implemented in software, firmware, or hardware;

   e) Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;

   f) Brief description of the IUT or the product/product family in which the IUT is implemented by the vendor (2-3 sentences);

2. Configuration information for the RSASP1VS tests

   a) RSA signature schemes supported:

      i. RSA_SSA-PKCS1-v1_5

      ii. RSA_SSA-PSS

   b) If RSA_SSA-PKCS1-v1_5 supported, indicate which hash algorithms are supported:

i. SHA1

ii. SHA224

iii. SHA256

6.2 The RSASP1 Test for RSA_SSA-PKCS1-v1_5

The RSASP1 test for testing the RSA_SSA-PKCS1-v1_5 algorithm generates one request (.req) file: *SigGenComponent15_186-3.req*.

Each file begins with a header that has the CAVS version on line one, the name of the test and the RSA algorithm being tested - "FIPS186-3 – SigGenComponent RSA PKCS#1 Ver 1.5" and the implementation name on line two, and the modulus size and a list of the hash algorithms tested on line three. Line 4 displays the date the file was generated. Line 5 is the mod size being tested in square brackets. This is always mod = 2048 for this component test.

```
# CAVS 14.1
# "FIPS186-3 - SigGenComponent RSA PKCS#1 Ver 1.5" information for "test1.5"
# Combination tested: Mod Size 2048 with SHA1  with SHA224  with SHA256

# Generated on Tue Oct 09 10:47:19 2012

[mod = 2048]
```

For each SHA selected, CAVS generates 30 trials. Each trial, or count, has the following format:

```
COUNT = 0
SHAAlg = SHA1
Msg =
0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff003021300906052b
0e03021a0500041477ed3ab6d0c44ef2f02817f9f411bd2363392394

COUNT = 1
SHAAlg = SHA1
Msg =
0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff003021300906052b
0e03021a0500041406045b10b4ad82733f2edc1590da3b0bb597c27d
```

SHAAlg is the hash algorithm being used in this test. This is used to determine the size of the hashed message and the DER code associated with the hash algorithm for use in the encoded message. Msg is the encoded message in the format for PKCS1-v1_5 as specified in PKCS #1 v2.1 Section 9.2 EMSA-PKCS1-v1_5. The Msg value is represented in hexadecimal. The sample file indicates the outputs that the IUT needs to provide.

```
[mod = 2048]

n = ?

e = ?
COUNT = 0
SHAAlg = SHA1
Msg =
0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff003021300906052b
0e03021a0500041477ed3ab6d0c44ef2f02817f9f411bd2363392394
S = ?

COUNT = 1
SHAAlg = SHA1
Msg =
0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff003021300906052b
0e03021a05000414406045b10b4ad82733f2edc1590da3b0bb597c27d
S = ?

.
.
.

n = ?

e = ?
COUNT = 30
SHAAlg = SHA224
Msg =
0001ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffffffffffffffffffffff00302d300d06096086480165030402040500041c
4b7e24e43d5abae2418e952c182e8b0a77681ecdfcae09106518193e
S = ?
```

The n value is the modulus value used in generating the IUT's public/private key pair. The e value is the public key of the IUT. The S value is the resulting signature from performing the exponentiation equation S = Msg^{d} mod n in the RSASP1 function.

The RSASP1VS verifies the correctness of the IUT's value of S by applying the exponentiation equation Msg = S^e mod n as specified in the RSAVP1 function (Section 5.2.2 of PKCS#1 v2.1). If the resulting Msg is the same as the Msg supplied by the CAVS tool for that trial, then the IUT's implementation of the RSASP1 function passes the validation test. If the values do not match, the IUT has an error in it. If an error occurs during validation, the values computed by the CAVS and the IUT are written to the log file. The CST Laboratory can use the information in the log file to assist the vendor in debugging the IUT.

6.3 The RSASP1 Test for RSA_SSA-PSS

The RSASP1 test for testing the RSA_SSA- PSS algorithm generates one request (.req) file: *SigGenComponentPSS_186-3.req*.

Each file begins with a header that has the CAVS version on line one, the name of the test and the RSA algorithm being tested - "FIPS186-3 – SigGenComponent RSA PKCS#1 RSASSA-PSS" and the implementation name on line two, and the modulus size tested on line three. Line 4 displays the date the file was generated. Line 5 is the mod size being tested in square brackets. This is always mod = 2048 for this component test.

```
# CAVS 14.1
# "FIPS186-3 - SigGenComponent RSA PKCS#1 RSASSA-PSS" information for
"testPSS"
# Combination tested: Mod Size 2048
# Generated on Thu Oct 11 09:06:27 2012

[mod = 2048]
```

CAVS generates 30 trials. Each trial, or count, has the following format:

```
COUNT = 0
n =
a63fb6b665165b254ed49b84bfdb1912d900eb55d302a649c55a5640533c4bc22ace842e2ff7
d396ddca4ac226bcae5d390163c2b1599e81aa736a9fa0fad3ed006efd0666769988c99753c9
2882c4cefcd0586dd0c7fb01027225cbcdb6a5638dd414ee69b9db1a4ce3089349b8c83ce7e8
4da0e7073351100f64a738c999f11ccb6276d2f67bd199bbd31f2d5cdfe8155edd0e2733e8a3
24116ca535c622e788334e75911dd79e88da82655522e82ed42d5f4c7b78f0ee5ea6beb26fb7
18f7df1408da7d4051c24e4cb7e0f4ddcd6bf98039eacd92d02217b2ad8dcbab196c0799f79e
352a487626f389cd180075d8a8d1a59161692675499c1c65e14f3fe5
d =
1635d1dfa93ea4dba59df2cef7e0ba07521574db48ef042f3bddf742edbbd2f53449d5cfe3d9
ac9b6db30e6cc4c715565ffcc70aa62dee66ad52710eb56f7d2b9f10b4de0b8751b8bc11eb00
2758dd19381e4f8a1047ff4921be053da6947da100bc3235adcb4631cbced300f66ae8d97634
0b56f1367d8d1963ad13481b6ae4db90cfddd45f20148aefe1462271e484d9a8e5ece9b7dae5
58c5467d37869cf43e7ac7b10bd89825743b1c3ad10a25012dee4fadc23a5b277dd083e11bc4
0e40a035dffa2b44af2affafc7941448349a3ab1abc3cbcb584a900c8bffdfd5a077475dca2e
0d52e7e70863f86e190eddfbb1837b2075db28d2c561b7957e95894b
EM =
ff3fb6b665165b254ed49b84bfdb1912d900eb55d302a649c55a5640533c4bc22ace842e2ff7
d396ddca4ac226bcae5d390163c2b1599e81aa736a9fa0fad3ed006efd0666769988c99753c9
2882c4cefcd0586dd0c7fb01027225cbcdb6a5638dd414ee69b9db1a4ce3089349b8c83ce7e8
4da0e7073351100f64a738c999f11ccb6276d2f67bd199bbd31f2d5cdfe8155edd0e2733e8a3
24116ca535c622e788334e75911dd79e88da82655522e82ed42d5f4c7b78f0ee5ea6beb26fb7
```

```
18f7df1408da7d4051c24e4cb7e0f4ddcd6bf98039eacd92d02217b2ad8dcbab196c0799f79e
352a487626f389cd180075d8a8d1a59161692675499c1c65e14f40bc
```

CAVS supplies the modulus n, the private key d, and the embedded message EM in the format for PKCS1-PSS as specified in PKCS #1 v2.1 Section 9.1.1 EMSA-PSS. The values are represented in hexadecimal. The sample file indicates the outputs that the IUT needs to provide.

```
[mod = 2048]

COUNT = 0
n =
a63fb6b665165b254ed49b84bfdb1912d900eb55d302a649c55a5640533c4bc22ace842e2ff7
d396ddca4ac226bcae5d390163c2b1599e81aa736a9fa0fad3ed006efd0666769988c99753c9
2882c4cefcd0586dd0c7fb01027225cbcdb6a5638dd414ee69b9db1a4ce3089349b8c83ce7e8
4da0e7073351100f64a738c999f11ccb6276d2f67bd199bbd31f2d5cdfe8155edd0e2733e8a3
24116ca535c622e788334e75911dd79e88da82655522e82ed42d5f4c7b78f0ee5ea6beb26fb7
18f7df1408da7d4051c24e4cb7e0f4ddcd6bf98039eacd92d02217b2ad8dcbab196c0799f79e
352a487626f389cd180075d8a8d1a59161692675499c1c65e14f3fe5
d =
1635d1dfa93ea4dba59df2cef7e0ba07521574db48ef042f3bddf742edbbd2f53449d5cfe3d9
ac9b6db30e6cc4c715565ffcc70aa62dee66ad52710eb56f7d2b9f10b4de0b8751b8bc11eb00
2758dd19381e4f8a1047ff4921be053da6947da100bc3235adcb4631cbced300f66ae8d97634
0b56f1367d8d1963ad13481b6ae4db90cfddd45f20148aefe1462271e484d9a8e5ece9b7dae5
58c5467d37869cf43e7ac7b10bd89825743b1c3ad10a25012dee4fadc23a5b277dd083e11bc4
0e40a035dffa2b44af2affafc7941448349a3ab1abc3cbcb584a900c8bffdfd5a077475dca2e
0d52e7e70863f86e190eddfbb1837b2075db28d2c561b7957e95894b
EM =
ff3fb6b665165b254ed49b84bfdb1912d900eb55d302a649c55a5640533c4bc22ace842e2ff7
d396ddca4ac226bcae5d390163c2b1599e81aa736a9fa0fad3ed006efd0666769988c99753c9
2882c4cefcd0586dd0c7fb01027225cbcdb6a5638dd414ee69b9db1a4ce3089349b8c83ce7e8
4da0e7073351100f64a738c999f11ccb6276d2f67bd199bbd31f2d5cdfe8155edd0e2733e8a3
24116ca535c622e788334e75911dd79e88da82655522e82ed42d5f4c7b78f0ee5ea6beb26fb7
18f7df1408da7d4051c24e4cb7e0f4ddcd6bf98039eacd92d02217b2ad8dcbab196c0799f79e
352a487626f389cd180075d8a8d1a59161692675499c1c65e14f40bc
S = ? (If function fails, indicate S = FAIL)
```

The S value is the resulting signature from performing the exponentiation equation $S = Msg^d \bmod n$ in the RSASP1 function. If the RSASP1 function fails, indicate S = FAIL. The RSASP1 function would fail if the value for EM is larger than the value for n-1. The encoded message for PKCS1-PSS would allow this scenario to occur. If it does, the first step in the RSASP1 function which states

1. If the message representative m is not between 0 and n-1, output “message representative out of range” and stop.

would cause the signature to fail.

The RSASP1VS verifies the correctness of the IUT’s value of S by comparing it to previously computed CAVS values. If the resulting signatures match, then the IUT’s implementation of the RSASP1 function passes the validation test. If the values do not match, the IUT has an error in it. If an error occurs during validation, the values computed by the CAVS and the IUT are written to the log file. The CST Laboratory can use the information in the log file to assist the vendor in debugging the IUT.

Appendix A References

[1] PKCS#1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002.

[2] FIPS186-3, *Digital Signature Standard (DSS)*, FIPS Publication 186-3, National Institute of Standards and Technology, June 2009.

[3] *Security Requirements for Cryptographic Modules*, FIPS Publication 140-2, National Institute of Standards and Technology, May 2001.