The RSADP Primitive Component Validation System (RSADPVS)

Updated: June 16, 2014
Original: September 11, 2013

Sharon S. Keller

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

Update Log

6/16/14

- With transition removing 1024 mod size, 1024 mod size should be removed from this test.

1 Introduction

This document, *RSADP Primitive Component Validation System (RSADPVS),* specifies the procedures involved in validating implementations of the RSADP decryption operation specified in Section 7.1.2 of the *NIST SP 800-56B: Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography,* dated August 2009[1] and in Section 5.1.2 of the *PKCS#1 v2.1: RSA Cryptography Standard (June 14, 2002)* [2]. This primitive is used by the key transport scheme referenced in both [1] and [2]. The validation testing of the RSADP primitive validates the correctness of the basic decryption operation.

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

A set of RSADP primitive test vectors 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 RSADP primitive. The RSADP primitive performs the modular exponentiation function. In applications using PIV cards, this exponentiation function is performed on the card to recover the key being transported. The RSADPVS provides testing to determine the correctness of the implementation of the primitive.

3 Conformance

The successful completion of the individual validation test(s) contained within the RSADPVS is required to claim conformance to the RSADP 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 RSADPVS
IUT	Implementation Under Test
PKCS	Public Key Cryptography Standards

4.2 Abbreviations

ABBREVIATION	MEANING
ANS	American National Standard
FIPS	Federal Information Processing Standard
IUT	Implementation Under Test
RSADP	RSA Decryption Primitive
RSADPVS	RSADP Primitive Validation System
SP	Special Publication

5 Design Philosophy of the RSADP Primitive Validation System

The RSADPVS is designed to test conformance to specifications in Section 7.1.2 of [1] and Section 5.2 of [2] 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 RSADPVS has the following design philosophy:

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

2. The testing performed within the RSADPVS 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 RSADPVS Tests

When applied to an IUT, the RSADPVS provides testing to determine the correctness of the RSADP function. The RSADPVS consists of a single validation test.

6.1 Configuration Information

To initiate the validation process of the RSADPVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of the RSADP 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 RSADPVS 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);

6.2 The RSADP Test

The RSADP test generates one request (.req) file: *RSADPComponent800_56B.req*.

Each file begins with a header that has the CAVS Tool version on line one, the name of the test and the RSA algorithm being tested - "RSADP Component (from SP800-56B)" and the

implementation name on line two, and the modulus size(s) tested on line three. Line 4-9 specifies one requirement to the testing. Line 10 displays the date the file was generated.

```
# CAVS 14.4
# "RSADP Component (from SP800-56B)" information for "test4"
# Combination tested: Mod Size 2048
# REQUIREMENT:  At least 10 of each of the 30 sets of values for each mod
# size must exercise Step 1 of the RSADP: "If the ciphertext
# representative c is not between 0 and n-1, output "ciphertext
# representative out of range" and stop"
# If this requirement is not met, the test will fail and IUT should
# be asked to rerun with correct number of tests.

# Generated on Thu May 09 16:02:07 2013
[mod = 2048]
```

RSADPVS generates 30 ciphertext *c* values. The length of *c* is 2048 bits because the modulus size being tested is 2048. *c* is represented in hexadecimal.

Each trial, or count, has the following format:

```
COUNT = 0

c =
f93b2f18418629de11f292dabc547f1a28ce1876dd974c735bfac367b1a214de4b14393d0c7d40
a59dae421a046599e578ce4e367349d10e0a6eed5c7a4efa6703fc556bf94d0d8718dbdfd61850
65dd81e18a25c11841287bd77772239a4500c8fc6bec307dfa4c2a825057209d49c9ec0fb489ec
8b0489593d48eed40b5e72421b6306da956dbb7caa7de245d5af726060b4ab1f2b82f94aa94a1b
b72bab841a6125021f1dd09db65b4ec2ae7ffea186dd72d3807e2440f6216440f1819500f8d699
28269e8aba809dab3d85239e1c068dd749370e502507b07ea4ad9585ac8383a8bf18a003da8079
7c9def49f7ab8779c9546d636f72225b819011f0e152
```

The sample file indicates the outputs that the IUT need to provide.

```
[mod = 2048]

COUNT = 0
n = ?

e = ?
c =
f93b2f18418629de11f292dabc547f1a28ce1876dd974c735bfac367b1a214de4b14393d0c7d40
a59dae421a046599e578ce4e367349d10e0a6eed5c7a4efa6703fc556bf94d0d8718dbdfd61850
65dd81e18a25c11841287bd77772239a4500c8fc6bec307dfa4c2a825057209d49c9ec0fb489ec
8b0489593d48eed40b5e72421b6306da956dbb7caa7de245d5af726060b4ab1f2b82f94aa94a1b
b72bab841a6125021f1dd09db65b4ec2ae7ffea186dd72d3807e2440f6216440f1819500f8d699
28269e8aba809dab3d85239e1c068dd749370e502507b07ea4ad9585ac8383a8bf18a003da8079
7c9def49f7ab8779c9546d636f72225b819011f0e152

Result = ? (Pass or Fail)

k (only if Result = Pass; Otherwise, remove this line) = ?
```

The IUT supplies each of these values for each of the 30 trials:

- The *n* value is the modulus value used in generating the IUT's public/private key pair. The modulus *n* shall be different for each trial.
- The *e* value is the public key of the IUT.
- The *c* value is the ciphertext from the request file. (NOTE: It is supplied by the RSADPVS. It is not generated by the IUT.)
- The *Result* line returns *Pass* or *Fail.* The *Result* will fail if the *c* does not satisfy the first step in the RSADP function which states:
    - If the ciphertext *c* is not such that $1 < c < n - 1$, output an indication that the ciphertext is out of range and stop.

  In order to test this condition, the IUT shall submit 10 out of 30 trials that trigger this failure. That is, 10 out of 30 trials where the *n* supplied by the IUT is less than or equal to the *c+1* supplied by the RSADPVS. The other 20 trials will *Pass*.
- The *k* value is the resulting keying material from performing the exponentiation equation $k = c^d \bmod n$ in the RSADP function. This line is only present if *Result = Pass*. Otherwise, the *k* line is removed.

The RSADPVS verifies the IUT's ability to determine if the values in a dataset should pass or fail. If the trial's *Result = Fail*, the RSADPVS checks to assure the failure was caused by *c* being out of range. If *c* is not out or range but the IUT indicates *Result = Fail*, the IUT has an error in it.

If the dataset passes, the RSADPVS verifies the correctness of the IUT's value of *k* by applying the exponentiation equation $k = c^d \bmod n$ as specified in the RSAVP1 function. If the resulting *k* is the same as the *k* supplied by the CAVS tool for that trial, then the IUT's implementation of the RSADP function passes the validation test. If the values do not match, the IUT has an error in it.

The values computed by the CAVS and the IUT are written to the log file if an error occurs during validation. The CST Laboratory can use the information in the log file to assist the vendor in debugging the IUT.

Appendix A References

[1] Special Publication 800-56B, *Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*, National Institute of Standards and Technology, August 2009.

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

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