The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS)

Updated: March 18, 2014
Previously Updated: September 17, 2013
Original: May 27, 2010

Timothy A. Hall

Sharon S. Keller

National Institute of Standards and Technology

Information Technology Laboratory

TABLE OF CONTENTS

Update Log

3/18/14

- Updated Section 6.2, Key Pair Generation, to indicate the three key pair generation methods supported. Two are from FIPS 186-4, and the third is from Section A.4.3 of ANSI X9.62-2005, only to be used for revalidation of FIPS 186-2 ECDSA implementations.

9/17/13

- Remove word “partial” from Signature Generation Component section.

9/5/13

- Updated References to FIPS 186-4 throughout document.

1/9/13

- Added Section 6.4.1 Partial Signature Generation Component Test.

1/17/12

- Section 6.2
  - Corrected key pair generation test description.

6/29/11

- Section 2
  - Removed second paragraph on testing of prerequisites.
- Section 3
  - Removed references to SHAVS and DRBGVS.
- Section 4.2
  - Removed DRBG, DRBGVS, SHA and SHAVS from list of abbreviations.

- Section 6
    - Removed prerequisites chart.

8/30/10

- Section 2 Last Paragraph
    - Correct the prerequisite testing to indicate Signature Verification only requires SHA.
- Section 6
    - Correct word o to “on”.
    - Change “to obtained” to “to be obtained”.
    - Change Section XX to 6.1 #8.
    - Update Chart to remove DRBG from Signature Verification.

1 Introduction

The Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS) specifies the procedures involved in validating implementations of the Elliptic Curve Digital Signature Algorithm (ECDSA) as approved in FIPS 186-4, *Digital Signature Standard (DSS)*[1] and specified in ANSI X9.62-2005, *Public Key Cryptography for Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*[2]. The ECDSAVS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the ECDSA2VS.

This document provides the basic design and configuration of the ECDSA2VS. Included are the specifications for testing the individual ECDSA components of the IUT. These components are:

- Key Pair Generation,
- Public Key Validation,
- Signature Generation, and
- Signature Verification.

In addition to providing validation testing for implementations of the complete Signature Generation function as specified in FIPS186-4, ECDSA2VS also provides validation testing for implementations of the Signature Generation function that assumes the input message is already hashed. This partial component validation test is designed for applications such as the PIV card. IUTs that test this partial Signature Generation function will receive a Component Validation.

This document defines the purpose, the design philosophy, and the high-level description of the validation process for ECDSA. The requirements and administrative procedures to be followed by those seeking formal validation of an implementation of ECDSA are presented. The requirements described include the specification of the data communicated between the IUT and the ECDSA2VS, the details of the tests that the IUT must pass for formal validation, and general instruction for interfacing with the ECDSA2VS.

2 Scope

This document specifies the tests required to validate IUTs for conformance to the ECDSA as specified in [1]. When applied to IUTs that implement ECDSA, the ECDSA2VS provides testing to determine the correctness of the algorithm components contained in the implementation. The ECDSA2VS is composed of a separate validation test to validate each of the various complete algorithm components and one test to validate the partial Signature Generation component that assumes the input message has already been hashed. In addition to determining conformance to the cryptographic specifications, the ECDSA2VS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the ECDSA implementation.

3 Conformance

The successful completion of the applicable tests contained within the ECDSA2VS is required to be validated as conforming to the FIPS PUB 186-4 ECDSA. Testing for the cryptographic module in which the ECDSA 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 ECDSA2VS
Elliptic Curve Digital Signature Algorithm	The algorithm specified in FIPS 186-4, *Digital Signature Standard (DSS)* for generating and verifying digital signatures.

4.2 Abbreviations

ABBREVIATION	**MEANING**
ECDSA	Elliptic Curve Digital Signature Algorithm specified in FIPS 186-4
ECDSA2VS	FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System
FIPS	Federal Information Processing Standard
IUT	Implementation Under Test
PKV	Public Key Validity

5 Design Philosophy of the Elliptic Curve Digital Signature Algorithm Validation System

The ECDSA2VS is designed to test conformance to ECDSA 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 ECDSA2VS has the following design philosophy:

1. The ECDSA2VS is designed to allow the testing of an IUT at locations remote to the ECDSA2VS. The ECDSA2VS and the IUT communicate data via *REQUEST (.req)* and *RESPONSE (.rsp)* files.

2. The testing performed within the ECDSA2VS 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 standard.

6 ECDSA2VS Tests

The ECDSA2VS for ECDSA consists of separate tests for each of four distinct components of ECDSA and one test for the partial testing of the Signature Generation component. The ECDSA2VS provides conformance testing for each of the components of the algorithm, as well as testing for apparent implementation errors. The components tested are:

- Key Pair Generation
- Public Key Validity (PKV)
- Signature Generation
- Signature Validation
- Signature Generation Component expecting hashed message as input (assumes hashing of the message already done)

6.1 Configuration Information

To initiate the validation process of the ECDSA2VS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of ECDSA. 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 ECDSA2VS to perform the specific tests. More specifically, the request for validation includes:

1. Vendor Name;
2. Product Name;
3. Product Version;
4. Implementation in software, firmware, or hardware;
5. Processor and Operating System with which the IUT was tested if the IUT is implemented in software or firmware;

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

7. The NIST curves and SHA sizes (e.g., SHA-256) supported by the IUT.

8. Indicate if testing the partial ECDSA Signature Generation component

6.2 Key Pair Generation Test

Key pairs for the ECDSA consist of pairs (*d*, *Q*), where the private key, *d*, is an integer, and the public key, *Q*, is an elliptic curve point. There are three methods by which these pairs may be generated. Two are from FIPS 186-4, sections B.4.1 and B.4.2, respectively. The third is the technique described in Section A.4.3 of ANSI X9.62-2005. This third technique is only used for adding a new operating environment (OE) to an existing FIPS 186-2 ECDSA validated implementation.

The ECDSA2VS tests the generation of key pairs for correctness by having the IUT produce 10 key pairs for each one of the three methods selected. The format and steps are the same for all three. The private key provided is used to compute the public key, *Q'*. The computed value *Q'* is then compared to the supplied public key, *Q*.

The ECDSA2VS:

A. Creates a *REQUEST* file (Filename: KeyPair.req) containing:

   1. The Product Information (vendor, product name, version);

   2. An indication of the NIST Recommended Curve(s) supported.

   Note: The CST laboratory sends the *REQUEST* file to the IUT.

The IUT:

A. Generates the key pairs specified in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: KeyPair.rsp) containing:

   1. The Product Name;

   2. For each curve supported the ten key pairs consisting of:

      a. The private key, *d*,

      b. The public key, *Q*.

   Note: The IUT sends the *RESPONSE* file to the CST laboratory for processing by the ECDSA2VS.

The ECDSA2VS:

A. Recalculates the public key, *Q'*, from the private key supplied in the response file. The value *Q'* is then compared to the supplied value *Q*.

B. If all values of *Q'* match the supplied values of *Q*, records PASS for this test; otherwise, records FAIL.

6.3 Public Key Validation Test

An IUT may include a routine for EC public key validation, as specified in Section A.4.2 of ANSI X9.62-2005. If so, the ECDSA2VS will generate 12 key pairs for each supported curve, modify some of the public keys to introduce errors, and determine whether or not the IUT can detect these errors.

The ECDSA2VS:

A. Generates 12 sets of valid key pairs for each curve supported by the IUT.

B. Makes a copy of the valid public keys created above, modifies some of the public keys to introduce errors by changing the value of the key, and runs Public Key Validation on the modified keys to be sure that the modification does not inadvertently result in a valid key.

C. Creates a *REQUEST* file (Filename: PKV.req) containing:

1. The Product Information (vendor, product name, version); and

2. For each curve supported, the 12 public keys from step B above.

Note: The CST laboratory sends the *REQUEST* file to the IUT.

D. Creates a *FAX* file (Filename: PKV.fax) containing:

1. The information from the *REQUEST* file; and

2. For each public key, an indication of whether the key should pass the Public Key Validation test.

Note: The CST laboratory retains the *FAX* file.

The IUT:

A. For each public key found in the *REQUEST* file, determines whether or not the public key passes all the conditions in Section A.4.2 of ANSI X9.62.

B. Creates a *RESPONSE* file (Filename: PKV.rsp) containing:

1. The information from the *REQUEST* file; and

2. For each public key, an indication of whether the key passes or fails the Public Key Validation test.

Note: The IUT sends the *RESPONSE* file to the CST laboratory for processing by the ECDSA2VS.

The ECDSA2VS:

A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.

B. If the results for all public keys match, records PASS for this test; otherwise, records FAIL.

6.4 Signature Generation Test

An implementation of the ECDSA may generate the (*r*,*s*) pairs that represent a digital signature. This option tests the ability of an IUT to produce correct signatures. To test signature generation, the ECDSA2VS supplies ten messages to the IUT. The IUT generates the corresponding signatures and returns them to the ECDSA2VS. The ECDSA2VS validates the signatures by using the associated public key to verify the signature.

The ECDSA2VS:

A. Creates a *REQUEST* file (Filename: SigGen.req) containing:

1. The Product Name;
2. For each modulus size and SHA length supported, ten messages to be signed.

Note: The CST laboratory sends the *REQUEST* file to the IUT.

The IUT:

A. Generates the signatures for the messages supplied in the *REQUEST* file.

B. Creates a *RESPONSE* file (Filename: SigGen.rsp) containing:

1. The Product Name;
2. The Domain Parameters used to sign the messages;
3. The messages that are signed;
4. The public key, *y*, corresponding to the private key, *x*, used to generate the signature; and
5. For each message, the computed signature values, *r* and *s*.

Note: The IUT sends the *RESPONSE* file to the CST laboratory for processing by the ECDSA2VS.

The ECDSA2VS:

A. Uses the respective public keys to verify the signatures in the *RESPONSE* file.

B. If all conditions are met, records PASS for this test; otherwise, records FAIL.

6.4.1 Signature Generation Component Test

This component validation test validates the ECDSA Signature Generation Test but assumes the input message has already been hashed. It is used by implementations such as the PIV card. The validation testing for the Signature Generation Component is the same as the ECDSA Signature Generation Test described in Section 6.4 except the ten messages that are supplied by the

ECDSA2VS represent already hashed messages and are therefore the length of the hash value being tested. When the IUT generates the signature on these hashed messages, it will bypass the hash step in the ECDSA Signature Generation function but will perform the rest of the function as described in the standard. The ECDSAVS will then use the respective public keys to verify the signatures in the *RESPONSE* file as described in Section 6.4 above.

6.5 Signature Verification Test

This option tests the ability of the IUT to recognize valid and invalid signatures. For each mod size selected, the ECDSA2VS generates a key pair, (*x*, *y*), of which the private key *x* is used to sign 15 pseudorandom messages of 1024 bits. Some of the messages or signatures are altered so that signature verification should fail. The messages, signatures, domain parameters, and public key *y* values are then forwarded to the IUT. The IUT then attempts to verify the signatures and returns the results to the ECDSA2VS, which compares the received results with its own stored results.

The ECDSA2VS:

A. For each of the supported modulus size and SHA length generates 15 sets of the following information:

   1. A pseudorandom message,

   2. A public/private key pair, and

   3. A signature for the message using the private key.

B. For approximately half of the message/signature sets, alter either the message, the public key, or the signature such that the message verification fails.

C. Creates a *REQUEST* file (Filename: SigVer.req) containing:

   1. The Product Name;
   2. Domain parameters for the supported modulus size,
   3. The information from step B, including:
      a. The pseudorandom message,
      b. A public key corresponding to the private key used to sign the messages, and
      c. The signature components *r* and *s*.

   Note: The CST laboratory sends the *REQUEST* file to the IUT.

D. Creates a *FAX* file (Filename: SigVer.fax) containing:

   1. The information from the *REQUEST* file; and

2. For each message/public key/signature set, an indication of whether the signature verification process should pass or fail. (Note: The SigVer.fax file also contains the private key used to create the original signature.)

The IUT:

A. Attempts to verify the signatures for the messages supplied in the *REQUEST* file using the corresponding domain parameters and public key.

B. Creates a *RESPONSE* file (Filename: SigVer.rsp) containing:

1. The information from the *REQUEST* file;

2. For each message/public key/signature set, an indication of whether the signature verification passed or failed.

Note: The IUT sends the *RESPONSE* file to the CST laboratory for processing by the ECDSAVS.

The ECDSA2VS:

A. Compares the contents of the *RESPONSE* file with the contents of the *FAX* file.

B. If the results for all message/public key/signature sets match, records PASS for this test; otherwise, records FAIL.

Appendix A References

[1] *Digital Signature Standard (DSS)*, FIPS Publication 186-4, National Institute of Standards and Technology, July 2013.

[2] *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, ANS X9.62-2005, American National Standard for Financial Services, November 16, 2005.

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