The Digital Signature Algorithm Validation System (DSAVS)

July 1, 2009

Timothy A. Hall

National Institute of Standards and Technology

Information Technology Laboratory

Computer Security Division

TABLE OF CONTENTS

1 Introduction

This document, *The Digital Signature Algorithm Validation System (DSAVS),* specifies the procedures involved in validating implementations of the Digital Signature Algorithm as approved in FIPS 186-3, *Digital Signature Standard (DSS)* [1]. FIPS 186-3 supports key sizes greater than or equal to 1024 bits by specifying four choices for the pair L and N (i.e., the bit lengths of the prime modulus *p* and the prime divisor of *(p-1)*, *q*, respectively). These four (L, N) pairs are (1024, 160), (2048, 224), (2048, 256), and (3072, 256). It also specifies a secure hash algorithm (SHA) of security strength greater than or equal to the security strength of the (L, N) pair. The DSAVS is designed to perform automated testing on Implementations Under Test (IUTs). This document provides the basic design and configuration of the DSAVS. Included are the specifications for testing the individual DSA components of the IUT. These components are:

- Domain Parameter Generation,
- Domain Parameter Verification,
- Key Pair Generation,
- Signature Generation, and
- Signature Verification.

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

2 Scope

This document specifies the tests required to validate IUTs for conformance to the DSA as specified in [1]. When applied to IUTs that implement DSA, the DSAVS provides testing to determine the correctness of the algorithm components contained in the implementation. The DSAVS is composed of five separate tests, one to validate each of the various algorithm components. In addition to determining conformance to the cryptographic specifications, the DSAVS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the DSA implementation.

3 Conformance

The successful completion of the tests contained within the DSAVS is required to be validated as conforming to the DSA. Testing for the cryptographic module in which the DSA is implemented is defined in FIPS PUB 140-2, *Security Requirements for Cryptographic Modules*[2].

4 Definitions and Abbreviations

4.1 Definitions

DEFINITION	**MEANING**
CMT laboratory	Cryptographic Module Testing laboratory that operates the DSAVS
Digital Signature Algorithm	The algorithm specified in FIPS 186-3, *Digital Signature Algorithm (DSA)* for generating and verifying digital signatures.

4.2 Abbreviations

ABBREVIATION	**MEANING**
DSA	Digital Signature Algorithm specified in FIPS 186-3
DSAVS	Digital Signature Algorithm Validation System
IUT	Implementation Under Test

5 Design Philosophy of The Digital Signature Algorithm Validation System

The DSAVS is designed to test conformance to DSA 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 DSAVS has the following design philosophy:

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

2. The testing performed within the DSAVS 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 DSAVS Tests

The DSAVS for DSA consists of separate tests for each of five distinct components of DSA. The DSAVS provides conformance testing for each of the components of the algorithm, as well as testing for apparent implementation errors. The components tested are:

- Domain Parameter Generation
- Domain Parameter Validation
- Key Pair Generation
- Signature Generation
- Signature Validation

6.1 Configuration Information

To initiate the validation process of the DSAVS, a vendor submits an application to an accredited laboratory requesting the validation of its implementation of DSA. 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 DSAVS 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 (L, N) pairs and SHA sizes (e.g., SHA-256) supported by the IUT.

8. If the IUT only handles specific values of P,Q, and G, these must be supplied to the CMT lab.

6.2 The Domain Parameter Generation Test

The domain parameters p and q must be generated either using the method of Appendix A.1.1.2 of FIPS 186-3, for probable primes, or A.1.2 of FIPS 186-3 for guaranteed primes. Currently, we only test the method in A.1.1.2 for generating probable primes. The generator g must be generated using either the method of Appendix A.2.1 of FIPS 186-3, for an unverifiable generation, or the method of A.2.3 for a verifiable canonical generation of the value. Currently, we only test the method in A.2.1. The algorithm used to generate the parameters produces as optional outputs, along with the domain parameters, a *domain_parameter_seed* value and a *counter* value.

The DSAVS tests the generation of domain parameters by asking the IUT to generate approximately five domain parameter sets for each modulus size selected by the vendor. This test verifies that the *SEED* value supplied results in the correct values for *p*, *q*, and *counter*. Additionally, the derived value *g* is consistent with the value of *h* returned by the IUT.

The DSAVS:

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

    1. The Product Name;

    2. The modulus and SHA size(s) supported; and

    3. The number of Domain Parameter sets to be generated for each mod size.

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

The IUT:

A. Generates the requested domain parameters specified in the *REQUEST* file.

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

    1. The Product Name;

    2. The modulus and SHA size(s) supported; and

    3. The following domain parameters generated by the IUT:

        a. *p* – the prime modulus,

        b. *q* – the prime divisor of *p*-1,

        c. *Seed* – the seed used to generate q,

        d. *c* – the value of the counter output from the generation of *p*,

        e. *g* – a group element of order *q*, and

f. *h* – the value used to generate g.

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the DSAVS.

The DSAVS:

A. Verifies that *SEED* produces the values of *p*, *q*, and *counter* using the algorithm in Appendix A.1.1.3 of FIPS 186-3, and that the value of *h* produces the value *g* as specified in Appendix A.2.2 of FIPS 186-3.

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

6.3 The Domain Parameter Validation Test

The prime parameters *p* and *q* must be generated by the method specified in Appendix A.1.1.2 of FIPS 186-3. Therefore, if an IUT accepts values of *p*, *q*, and *g* from an external source, the IUT assumes that Appendix A.1.1.2 and Appendix A.2.1 were used to generate those values. For each modulus size, the DSAVS supplies sextets (*SEED*, *q*, *p*, *counter*, *g*, *h*) to the IUT. Some of the values in some of the sextets are modified before being passed to the IUT. The IUT verifies the correctness of each sextet, and returns the results to the DSAVS, which compares these received results with its own stored results.

The DSAVS:

A. Generates five correct sets of domain parameter for each modulus sizes supported by the IUT. Each set of parameters contains:

1. *p* - the prime modulus,

2. *q* – the prime divisor of *p*-1,

3. *Seed* – the seed value used to generate *q*,

4. *c* – the value of the counter output from the generation of *p*,

5. *g* – a group element of order *q*, and

6. *h* – the value used to generate *g*.

B. Modify the valid domain parameter sets created above. One parameter from each of the sets is modified in the following manner:

1. Modify *p* such that the result is not prime,

2. Modify *q* such that it does not divide *p*-1;

3. Modify the *Seed*;

4. Modify *g* such that $g \neq h^{(p-1)/q}$ mod *p*; or

5. No modification is performed.

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

1. The Product Name; and

2. The domain parameter sets from step B, containing:

    a. *p* - the prime modulus,

    b. *q* – the prime divisor of *p*-1,

    c. *Seed* – the seed value used to generate *q*,

    d. *c* – the value of the counter output from the generation of *p*,

    e. *g* – a group element of order *q*, and

    f. *h* – the value used to generate *g*.

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

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

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

2. For each domain parameter set, an indication of whether the set should pass the domain parameter validation test.

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

The IUT:

A. For each domain parameter set found in the *REQUEST* file, verifies that the *SEED* provided generates the same set of domain parameters using the procedures found in Appendix A.1.1.2 and Appendix A.2.2 of FIPS 186-3.

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

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

2. For each domain parameter set, an indication of whether the set was properly regenerated.

Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the DSAVS.

The DSAVS:

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

B. If the results for all domain parameter sets match, records PASS for this test; otherwise, records FAIL.

6.4 Key Pair Generation Test

Key pairs for DSA consist of pairs *x* and *y*, the private and public key respectively. The private key is generated by the Random Number Generation (RNG) method specified in Appendix A.2.1 or A.2.3 of FIPS 186-3. Testing of the RNG method is performed with the RNGVS test that is an independent test outside of the DSAVS. In order to have the private key component validated the RNGVS must also be performed.

The DSAVS tests the generation of key pairs for correctness by having the IUT provide domain parameters, *p*, *q*, and *g*; and ten sets of private key, *x*, and public key, *y*, pairs. The DSAVS validates that the private key is in the proper range and the public key is derived from the private key.

The DSAVS:

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

   1. The Product Name; and
   2. The number of key pairs to be generated per mod and SHA size.

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

B. Creates a *FAX* file (Filename: KeyPair.fax) containing the information from the *REQUEST* file.

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

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 modulus size supported, the following information:
      a. Domain Parameters for the supported modulus size, and
      b. The requested number of sets of *x* and *y* values.

   Note: The IUT sends the *RESPONSE* file to the CMT laboratory for processing by the DSAVS.

The DSAVS:

A. Verifies that the *x* value is in the correct range ($0<x<q$), and that $y = g^x \bmod p$, as in Appendix A.2.2.

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

6.5 Signature Generation Test

An implementation of the DSA 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 DSAVS supplies ten messages to the IUT. The IUT generates the corresponding signatures and returns them to the DSAVS. The DSAVS validates the signatures by using the associated public key to verify the signature.

The DSAVS:

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

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

Note: The CMT 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 CMT laboratory for processing by the DSAVS.

The DSAVS:

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.

6.6 Signature Verification Test

This option tests the ability of the IUT to recognize valid and invalid signatures. For each mod size selected, the DSAVS 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 DSAVS, which compares the received results with its own stored results.

The DSAVS:

A. For each of the supported modulus size, 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 CMT 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 CMT laboratory for processing by the DSAVS.

The DSAVS:

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 186-3, National Institute of Standards and Technology, March 2006.

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