NIST SP 800-56A and its Revisions

by Swapneela Unkule, atsec CST Lab Manager

NIST SP 800-56A provides recommendations for asymmetric-key-based key-agreement schemes based on the Diffie-Hellman (DH) and Menezes-Qu-Vanstone (MQV) algorithms and is intended to provide sufficient information for a vendor to implement secure key establishment using asymmetric algorithms in a FIPS validated module.

SP 800-56A [56Ar1] was originally added to FIPS 140-2 [Annex D] on January 24, 2007. The second revision of SP 800-56A, denoted as [56Ar2], was published in May 2013, however it is only allowed to be claimed as a vendor-affirmed algorithm in a FIPS validated module, as there is no algorithm testing available to obtain CAVP certificates for this revision. There were several important differences between SP 800-56A Rev2 and SP 800-56A, the main one being that [56Ar2] permits the use of additional key derivation functions (KDF) documented in SP [800-135] and [SP 800-56C]. The third revision of SP 800-56A, denoted [56Ar3] was published in April 2018 and was added to FIPS 140-2 [Annex D] on June 10, 2019. There are several significant differences between SP 800-56A Rev3 and its predecessor, including the introduction of “safe-prime” groups and moving all of the key derivation functions to [SP 800-56C].
For simplicity and in the interest of FIPS validation, the differences listed here are between the original validation of SP 800-56A [56Ar1], which is the only version for which the algorithm testing is available as of now, and Revision 3 of SP 800-56A [56Ar3], which will be the only revision considered approved past December 31st 2020. Both [56Ar1] and [56Ar3] provide the same categories of key-agreement schemes based on the number of ephemeral keys used by the two parties in the key-agreement process; category 1 is based on “two ephemeral key pairs”, category 2 is based on “one ephemeral key pair” and category 3 is based on “zero ephemeral key pairs” i.e. using all static key pairs. [56Ar3] however includes additional checks for error handling in these categories. Next we will look into the differences with respect to each step of the key agreement process, namely domain parameter generation and validation, key generation and validation, shared secret computation, key derivation and key confirmation along with the required assurances.

We start with “Domain Parameter Generation”. The public and private key pairs used in the 56A key-establishment process need to be generated with respect to a set of domain parameters based on either Finite Field Cryptography (FFC) or Elliptic Curve Cryptography (ECC), depending on the key-agreement scheme used.

With respect to FFC domain parameter, the [56Ar1] only allowed the generation using a method specified in [FIPS 186-3] with a parameter-size set of either FA (public key size = 1024 bits & private key size = 160 bits), FB (public key size = 2048 bit & private key size = 224 bits) or FC (public key size = 2048 bit & private key size = 256 bits). [56Ar3] includes an additional class of domain parameter called safe-prime groups as approved for FFC based key agreement. The approved safe-prime groups are the ones employed by the “MODP” Diffie-Hellman groups used in the Internet Key Exchange protocol version 2 (IKE v2) listed in [RFC 3526] and the ones employed by the “ffdhe” Diffie-Hellman groups used in Transport Layer Security (TLS) protocol listed in [RFC 7919]. With respect to FIPS 186-type domain parameters, [56Ar3] removed the parameter-size set FA since it only provided a security strength of 80 bits and further clarified that FIPS 186-type domain parameters should only be used for backward compatibility with existing applications that cannot be upgraded to use the approved safe-prime groups.

With respect to ECC based domain parameter, [56Ar1] mandated generation of parameters as specified in [ANS X9.62] or selected from the recommended elliptic curve domain

parameters specified in [FIPS 186-3], which included parameter-size set of EA (curve sizes 160-223 bits), EB (curve sizes 224-255 bits), EC (curve sizes 256-383 bits), ED (curve sizes 384-511 bits) and EE (curve sizes 512+ bits). [56Ar3] only allows the ECC domain parameters selected from [SP 800-186], which are listed in its Appendix D, removing references to X9.62. The curves listed in Appendix D do not include a range of parameter-size sets. Instead it lists specific NIST-approved curves i.e. P curves with size 224, 256, 384, 512 and K or B curves with sizes 233, 283, 409, 571, removing the curve sizes below 224 bits that provided less than 112 bits of security strength. [56Ar3] also includes protocol-specific elliptic curves as approved for ECC key-agreement, which are defined in [RFC 4492] and [SP 800-52] used in TLS, and the ones defined in [RFC 5903] used in IPsec with IKE v2.

Please note that even though the SP 800-56A Rev3 was added to Annex D of FIPS 140-2 on June 10, 2019, both the safe-prime groups and the protocol-specific elliptical curves that are included in [56Ar3] were already allowed to be used in the FIPS-approved mode as per [140-2 IG D.13] published in 2017.

The next area to look at is “Domain Parameter Validation”. Secure key establishment depends on the arithmetic validity of the domain parameters used by the parties. Therefore, the 56A standard requires that each party shall have assurance of the validity of the domain parameters before they are used. The [56Ar1] specified three ways of achieving this assurance and required that at least one of them shall be used. First, a party generates the domain parameter itself, using approved method (as specified in above paragraph). Second, the party performs explicit domain parameter validation as per [FIPS 186-3] for FFC based parameters and as per [ANS X9.62] for ECC based parameters. Third, the party receives assurance from a trusted third party (for example, a certification authority (CA)). The [56Ar3] further refined the assurance methods for FFC based scheme by allowing the generation of domain parameters corresponding to approved safe-prime groups and by tightening the requirements for FIPS 186-type FFC domain parameters that mandate an explicit domain-parameter validation as specified in [SP 800-89] using the provided SEED and counter values.

Now we come to the “Key Generation” stage which includes requirements for the generation of key pairs to be used in the key agreement process, along with methods for obtaining assurances that valid key pairs are used. [56Ar3] explicitly states that the generated key pairs shall be used only for key-agreement purposes. For both FFC and ECC key pair generation methods [56Ar1] refers to Appendix B of [FIPS 186-3] i.e. the key generation methods using either “Extra Random Bits” or “Testing Candidates” methods which are exactly the same as key generation used under Digital signature schemes for DSA and ECDSA algorithms respectively. Key generation methods listed in [56Ar3] differ in the sense that they include slight modifications to the methods listed in the [FIPS 186-4] standard. The input parameters for the FFC Key generation methods in [56Ar3] include two additional values, namely the maximum bit length of the private key to be generated (N) and the maximum-security strength to be supported by the key pair. The [56Ar1] key generation process internally derives these values based on the input domain parameters while [56Ar3] requires these to be explicitly provided as an input to the key generation process. In the actual key generation process, [56Ar3] includes steps for error checking of these additional input parameters and it replaces every use of domain parameter ‘q’ with the min (2^N, q). [56Ar3] also differs in the length of the allowed range of sizes for the generated key pair. Specifically, the range for the private key has been changed from [1, q–1] to [1, min ($2^N - 1$, q− 1)] and from [1, p–1] to [2, p – 2] for the public key. Here ‘p’ and ‘q’ are part of the input FFC domain parameter. Similarly, ECC Key generation methods in [56Ar3] include one additional input parameter, namely the maximum-security strength(s) to be supported by the key pair. The [56Ar1] key generation process internally derived this value based on the input domain parameters while the [56Ar3] requires this to be explicitly provided as input to the key generation process. In the actual key generation process, the [56Ar3] includes additional steps for error checking of the additional input parameter(s) and also requires the non-approved domain parameter to be rejected, returning an error.

Prior to or during a key-agreement process, the participants in the transaction need to obtain the assurances about the key pairs used. The required assurance differs based on which entity is performing the assurance (i.e. the owner or the recipient of the key) and the type of key used (i.e. 'static' used in multiple transactions vs. 'ephemeral' used in exactly one transaction).

With respect to assurances required by the key pair owner, [56Ar1] only required assurances for possession of the private key and validity of the public key, whereas [56Ar3] mandates the owner to obtain assurances for correct generation of the key pair, possession of the key pair, assurance of the private as well as the public key validity, and assurance of pair-wise consistency using different methods listed in subsection of 5.6.2.1 of [56Ar3]. The assurance of the pair-wise consistency check can be obtained by the owner by either self-generating the static or ephemeral key using one of the approved methods listed in [56Ar3], or performing an explicit consistency check, which is only applicable for the static key pair generated by either the owner or a trusted third party. The vendors performing FIPS validation will be familiar with the term "pair-wise consistency", which is the process to verify that the generated key pair have a correct mathematical relationship by performing successful signature generation and verification using the generated keys. FIPS 140-2 requires this test to be performed on the asymmetric keys generated by the module. However, the assurance of pair-wise consistency specified in [56Ar3] is slightly different. Here the referenced assurance is obtained by re-computing the public key (from the private key and other domain parameters) and successfully comparing the computation result to the public key generated by the key generation process.

With respect to the recipient of the public key, the required types of assurances i.e. public key validation and private key possession are more or less the same in revision 1 and 3 of 56A, but the methods allowed for obtaining these assurances differ. For assurance of private key possession, [56Ar3] introduced an optional method for obtaining this assurance for ephemeral keys, which was absent in [56Ar1]. For assurance of the public key validation, one of the allowed methods in [56Ar1] for static public key validation includes obtaining the assurance that the trusted third party has generated the key pair, whereas for ephemeral public key validation, one of the allowed methods includes either full or partial public key validation using a trusted third party. [56Ar3] discontinued these two methods that relied on a trusted third party for obtaining assurance. For the actual process of performing public key validation [56Ar1] included full/partial key validation routines for ECC and only full key validation routine for FFC. The [56Ar3] retains those routines and also introduces an FFC partial public-key validation routine to be used for ephemeral keys generated using safe-prime groups only. Please note that for the public-key validation routines that are consistent between in [56Ar1] and [56Ar3], the required algorithm testing performed as part of a FIPS validation is much more stringent for [56Ar3] (using the new ACVP tool) than the testing performed for [56Ar1] (using the CAVS tool) in the past. The CAVS tool was retired in June 2020.

The next step in the key agreement process is to compute the shared secret, which is a secret value computed using a key-establishment scheme and is known by both participants. [56Ar3] includes the same type of DLC primitive allowed to be used as that of [56Ar1] namely FFC DH, ECC CDH, FFC MQV and ECC MQV. However, the procedure for calculating these primitives in [56Ar3] includes additional checks. Specifically, the calculation of FFC DH and FFC MQV primitives require an additional step to verify that the generated shared secret is greater than or equal to '1' and not equal to '(p-1)' where 'p' is part of the provided input FFC domain parameter. Additionally, all the four DLC primitive calculation procedures, include two additional steps not present in [56Ar1]. First, explicit steps for the destruction of all intermediate calculation values just before exiting from the routine after successful generation of the shared secret and in an event when an error is encountered. Second, instructions have been added for converting the shared secret value from 'integer to byte string' for FFC and from 'field-element to byte string' for ECC before outputting the shared secret value. [56Ar3] also includes a requirement that the shared

secret shall be used only by an approved key-derivation method, which is described in the next paragraph.

Next, we will take a look at the Key Derivation process that includes deriving key material using a shared secret. [56Ar1] included the Concatenation based KDF and its variant called the ASN.1 key derivation function. [56Ar3] references a stand-alone document [SP800-56C] for all the approved key derivation functions, which includes one-step and two-step KDFs as well as the application-specific KDFs listed in [SP800-135]. Please note that in order to use the [56Ar3] compliant key agreement in the FIPS-approved mode, the recently updated [140-2 IG] D.8 scenario X1 requires implementation of a self-test to cover the key derivation function in addition to the shared secret computation. This self-test requirement is different from the one required to claim compliance to [56Ar1]. Because in that case, the use of [800-56C] or [SP800-135] KDFs was considered as a non-approved but allowed key agreement method per scenario 3 of [140-2 IG] D.8, and there was no requirement to perform separate self-test on the KDFs per [140-2 IG] 9.6 as long as the underlying algorithms were tested.

The last step in the key agreement process is the Key Confirmation (KC) which refers to actions taken to provide assurance to one party (the key-confirmation recipient) that another party (the key-confirmation provider) possesses the correct secret keying material and/or the shared secret from which that keying material is derived. Key confirmation is considered successful if the 'MacTag' value computed by the key-confirmation recipient matches the 'MacTag' value that the recipient received from the key-confirmation provider. The calculation of a 'MacTag' value is performed by applying an approved MAC algorithm on the inputs 'MacKey' (which is a symmetric key derived using the shared secret that was computed by each party during the key-agreement) and certain context-specific 'MacData'. The [56Ar3] includes an additional approved hash function, SHA-3 as specified in [FIPS 202] and an additional approved MAC function, KMAC as specified in [SP800-185]. [56Ar3] requires the minimum MacTag length to be 64 bits. Additionally, for the MacKey length used by the HMAC and KMAC algorithms, [56Ar3] includes an upper bound of 512 bits and a lower bound equal to the supported security strength of the MAC algorithm used. [56Ar3] also included specific instructions to be followed when the key confirmation fails such as destruction of the derived key material by each participant and destruction of the key-agreement transaction. As per [140-2 IG] D.8 the testing of key confirmation step is only required if applicable i.e. if the module implements it. There is no self-test requirement for the key confirmation functionality.

In summary, here are the requirements to meet the [56Ar3] compliance under scenario X1 of IG D.8.

IG D.8 Scenario	Implementation Requirements from [56Ar3]	Testing Needed
X1 (1) KAS-SSC Modules only implementing DH/ECDH shared secret computation	- Shared secret computation per one of the schemes listed in section 6 and primitives in section 5.7. - Domain parameter selection/generation in section 5.5.1 and management per section 5.5.3. - Key generation (KeyGen) per section 5.6.1 and management per section 5.6.3 - Required assurances per section 5.5.2 and 5.6.2 - Use of approved DRBG per section 5.3 and Nonce requirement in section 5.4 (if applicable)	- CAVP certificate for SSC and KeyGen, DRBG (if applicable) - Known Answer test (KAT) for SSC - Pairwise-consistency Test for KeyGen
X1 (2) KAS Modules implementing DH/ECDH Key agreement	- All above requirements - KDF compliant to either SP800-56C Rev1 or 2 OR one of the KDFs listed in [140-2 IG] G.20 which includes SP 800-135 and TLS 1.3 KDF. - KC per section 5.9, if supported by the module.	- CAVP certificate for entire Key agreement or separate certs for SSC, KDF, SHA, HMAC, KC (if applicable).

	- Use of approved Hash per section 5.1and approved MAC per section 5.2 (if applicable)	- KAT for entire Key agreement or separate KATs for SSC, KDF. No KAT for KC. - CAVP certificate and Pairwise-consistency Test for KeyGen

References:

1. [56Ar1] https://csrc.nist.rip/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pdf
2. [56Ar2] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf
3. [56Ar3] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf
4. [140-2 IG] https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/fips1402ig.pdf
5. Annex D https://csrc.nist.gov/csrc/media/publications/fips/140/2/final/documents/fips1402annexd.pdf
6. [SP 800-56C] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr1.pdf
7. [SP 800-135] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-135r1.pdf
8. [RFC 8446] https://tools.ietf.org/html/rfc8446
9. [FIPS 186-3] https://csrc.nist.gov/csrc/media/publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf
10. [RFC 3526] https://tools.ietf.org/html/rfc3526
11. [RFC 7919] https://tools.ietf.org/html/rfc7919
12. [RFC 5903] https://tools.ietf.org/html/rfc5903
13. [RFC 4992] https://tools.ietf.org/html/rfc4492
14. [SP 800-89] https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-89.pdf
15. [SP 800-52] https://csrc.nist.gov/publications/detail/sp/800-52/rev-2/draft
16. ANS X9.62-2 Elliptic Curve Digital Signature Algorithm (ECDSA), November 16, 2005.