NIST Special Publication 800-133
Revision 1

Recommendation for Cryptographic Key Generation

Elaine Barker
Allen Roginsky

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-133r1

C O M P U T E R S E C U R I T Y

National Institute of Standards and Technology
U.S. Department of Commerce

NIST Special Publication 800-133
Revision 1

Recommendation for Cryptographic Key Generation

Elaine Barker
Allen Roginsky
Computer Security Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-133r1

July 2019

U.S. Department of Commerce
Wilbur L. Ross, Jr. Secretary

National Institute of Standards and Technology
Walter Copan, NIST Director and Under Secretary of Commerce for Standards and Technology

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3551 *et seq.*, Public Law (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including minimum requirements for federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

National Institute of Standards and Technology Special Publication 800-133 Revision 1
Natl. Inst. Stand. Technol. Spec. Publ. 800-133 Rev 1, 26 pages (July 2019)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-133r1

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Comments on this publication may be submitted to:

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930
Email: SP-800-133_Comments@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA)

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in Federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Abstract

Cryptography is often used in an information technology security environment to protect data that is sensitive, has a high value, or is vulnerable to unauthorized disclosure or undetected modification during transmission or while in storage. Cryptography relies upon two basic components: an algorithm (or cryptographic methodology) and a cryptographic key. This Recommendation discusses the generation of the keys to be managed and used by the **approved** cryptographic algorithms.

Keywords

asymmetric key; key agreement; key derivation; key generation; key wrapping; key replacement; key transport; private key; public key; symmetric key.

Acknowledgements

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by Rich Davis of the National Security Agency and the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Table of Contents

1 Introduction 1
2 Definitions, Acronyms and Symbols 1
2.1 Definitions 1
2.2 Acronyms 6
2.3 Symbols 7
3 General Discussion 8
3.1 Keys to Be Generated 8
3.2 Where Keys are Generated 8
3.3 Supporting a Security Strength 8
4 Using the Output of a Random Bit Generator 9
5 Generation of Key Pairs for Asymmetric-Key Algorithms 11
5.1 Key Pairs for Digital Signature Schemes 11
5.2 Key Pairs for Key Establishment 11
5.3 Distributing the Key Pairs 12
5.4 Key Pair Replacement 13
6 Generation of Keys for Symmetric-Key Algorithms 13
6.1 The “Direct Generation” of Symmetric Keys 14
6.2 Distributing the Generated Symmetric Key 14
6.3 Symmetric Keys Generated Using Key-Agreement Schemes 14
6.4 Symmetric Keys Derived from a Pre-shared Key 15
6.5 Symmetric Keys Derived from Passwords 15
6.6 Symmetric Keys Produced by Combining Multiple Keys and Other Data 16
6.7 Replacement of Symmetric Keys 17
Appendix A: References 18
Appendix B: Revisions 21

1 Introduction

Cryptography is often used in an information technology security environment to protect data that is sensitive, has a high value, or is vulnerable to unauthorized disclosure or undetected modification during transmission, or while in storage or while in use. Cryptography relies upon two basic components: an algorithm (or cryptographic methodology) and often, on a cryptographic key. The algorithm is a mathematical function, and the key is a parameter used by that function.

The National Institute of Standards and Technology (NIST) has developed a wide variety of Federal Information Processing Standards (FIPS) and NIST Special Publications (SPs) to specify and approve cryptographic algorithms for Federal government use. In addition, guidance has been provided on the management of the cryptographic keys to be used with these **approved** cryptographic algorithms.

This Recommendation discusses the generation of the keys to be used with the **approved** cryptographic algorithms. The keys are either generated using mathematical processing on the output of **approved** Random Bit Generators and possibly other parameters or generated based upon keys that are generated in this fashion.

2 Definitions, Acronyms and Symbols

2.1 Definitions

Approved	FIPS-approved and/or NIST-recommended.
Asymmetric key	A cryptographic key used with an asymmetric-key (public-key) algorithm. The key may be a private key or a public key.
Asymmetric-key algorithm	A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible. Also known as a public-key algorithm.
Bit string	An ordered sequence of 0 and 1 bits.
Ciphertext	Data in its encrypted form.
Compromise	The unauthorized disclosure, modification or use of sensitive data (e.g., keying material and other security-related information).
Cryptographic algorithm	A well-defined computational procedure that takes variable inputs, often including a cryptographic key, and produces an output.

Cryptographic boundary	An explicitly defined continuous perimeter that establishes the physical bounds of a cryptographic module and contains all the hardware, software and/or firmware components of a cryptographic module. See FIPS 140.
Cryptographic key (key)	A parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the correct key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples of cryptographic operations requiring the use of cryptographic keys include: 1. The transformation of plaintext data into ciphertext data, 2. The transformation of ciphertext data into plaintext data, 3. The computation of a digital signature from data, 4. The verification of a digital signature, 5. The computation of an authentication code from data, 6. The verification of an authentication code from data and a received authentication code, 7. The computation of a shared secret that is used to derive keying material. 8. The derivation of additional keying material from a key-derivation key (i.e., a pre-shared key).
Cryptographic module	The set of hardware, software, and/or firmware that implements security functions (including cryptographic algorithms and key generation) and is contained within a cryptographic module boundary. See FIPS 140.[1]
Cryptoperiod	The time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect.
Data integrity	A property possessed by data items that have not been altered in an unauthorized manner since they were created, transmitted or stored.
Decryption	The process of changing ciphertext into plaintext using a cryptographic algorithm and key.

[1] FIPS 140, *Security Requirements for Cryptographic Modules*.

Digital signature	The result of a cryptographic transformation of data that, when properly implemented, provides origin authentication, assurance of data integrity and supports signatory non-repudiation.
Encryption	The process of changing plaintext into ciphertext using a cryptographic algorithm and key.
Entity	An individual (person), organization, device or process. Used interchangeably with "party."
Entity authentication	The process of providing assurance about the identity of an entity interacting with a system (e.g., to access a resource). Sometimes called identity authentication.
Entropy	A measure of the disorder, randomness or variability in a closed system. See SP 800-90B.[2]
Key	See cryptographic key.
Key agreement	A (pair-wise) key-establishment procedure in which the resultant secret keying material is a function of information contributed by both participants, so that neither party can predetermine the value of the secret keying material independently of the contributions of the other party. Contrast with key transport.
Key-agreement primitive	A primitive algorithm used in a key-agreement scheme specified in SP 800-56A,[3] or in SP 800-56B.[4]
Key derivation	1. A process by which one or more keys are derived from a shared secret and other information during a key-agreement transaction. 2. A process that derives new keying material from a key (i.e., a key-derivation key) that is currently available.
Key-derivation key	A key used as an input to a key-derivation method to derive other keys. See SP 800-108.[5]
Key establishment	A procedure that results in secret keying material that is shared among different parties.
Key-generating module	A cryptographic module in which a given key is generated.
Key generation	The process of generating keys for cryptography.

[2] SP 800-90B, *Recommendation for the Validation of Entropy Sources for Random Bit Generation*.
[3] SP 800-56A, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*.
[4] SP 800-56B, *Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography*.
[5] SP 800-108, *Recommendation for Key Derivation Using Pseudorandom Functions*.

Key pair	A private key and its corresponding public key; a key pair is used with an asymmetric-key (public-key) algorithm.
Key-pair owner	In asymmetric-key cryptography, the entity that is authorized to use the private key associated with a public key, whether that entity generated the key pair itself, or a trusted party generated the key pair for the entity.
Key transport	A key-establishment procedure whereby one party (the sender) selects a value for the secret keying material and then securely distributes that value to another party (the receiver).
Key wrapping	A method of encrypting and decrypting keys and (possibly) associated data using symmetric-key cryptography; both confidentiality and integrity protection are provided. See SP 800-38F.[6]
Module	See Cryptographic module.
Origin authentication	A process that provides assurance of the origin of information (e.g., by providing assurance of the originator's identity).
Owner	1. For an asymmetric key pair, consisting of a private key and a public key, the owner is the entity that is authorized to use the private key associated with a public key, whether that entity generated the key pair itself or a trusted party generated the key pair for the entity. 2. For a symmetric key (i.e., a secret key), the entity or entities that are authorized to share and use the key.
Party	See Entity.
Password	A string of characters (letters, numbers and other symbols) that are used to authenticate an identity or to verify access authorization. A passphrase is a special case of a password that is a sequence of words or other text. In this document, the use of the term "password" includes this special case.
Permutation	An ordered (re)arrangement of the elements of a (finite) set; a function that is both a one-to-one and onto mapping of a set to itself.
Plaintext data	In this Recommendation, data that will be encrypted by an encryption algorithm or obtained from ciphertext using a decryption algorithm.

[6] SP 800-38F, *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*.

Pre-shared key	A secret key that has been established between the parties who are authorized to use it by means of some secure method (e.g., using a secure manual-distribution process or automated key-establishment scheme).
Private key	A cryptographic key used with an asymmetric-key (public-key) cryptographic algorithm that is not made public and is uniquely associated with an entity that is authorized to use it. In an asymmetric-key cryptosystem, the private key is associated with a public key. Depending on the algorithm that employs the private key, it may be used to: 1. Compute the corresponding public key, 2. Compute a digital signature that may be verified using the corresponding public key, 3. Decrypt data that was encrypted using the corresponding public key, or 4. Compute a key derivation key, which may then be used as an input to a key derivation process.
Public key	A cryptographic key used with an asymmetric-key (public-key) cryptographic algorithm that may be made public and is associated with a private key and an entity that is authorized to use that private key. Depending on the algorithm that employs the public key, it may be used to: 1. Verify a digital signature that is signed by the corresponding private key, 2. Encrypt data that can be decrypted by the corresponding private key, or 3. Compute a piece of shared data (i.e., data that is known only by two or more specific entities).
Public-key algorithm	See Asymmetric-key algorithm.
Random Bit Generator (RBG)	A device or algorithm that outputs bits that are computationally indistinguishable from bits that are independent and unbiased.
Rekey	A procedure in which a new cryptographic key is generated in a manner that is independent of the (old) cryptographic key that it will replace.
Secret key	A cryptographic key used by one or more (authorized) entities in a symmetric-key cryptographic algorithm; the key is not made public.

Secure channel	A path for transferring data between two entities or components that ensures confidentiality, integrity and replay protection, as well as mutual authentication between the entities or components. The secure channel may be provided using cryptographic, physical or procedural methods, or a combination thereof.
Security strength	A number associated with the amount of work (that is, the number of basic operations of some sort) required to break a cryptographic algorithm or system. Security strength is often expressed in bits. If the security strength is *S* bits, then it is expected that (roughly) 2^S basic operations are required to break the algorithm or system.
Shall	This term is used to indicate a requirement of a Federal Information Processing Standard (FIPS) or a requirement that must be fulfilled to claim conformance to this Recommendation. Note that **shall** may be coupled with **not** to become **shall not**.
Shared secret	A secret value that has been computed during an execution of a key-establishment scheme between two parties, is known by both participants, and is used as input to a key-derivation method to produce keying material.
Support a security strength	A term applied to a method (e.g., an RBG, or a key with its associated cryptographic algorithm) that is capable of providing (at a minimum) the security strength required or desired for protecting data.
Symmetric key	See Secret key.
Symmetric-key algorithm	A cryptographic algorithm that uses the same secret key for its operation and, if applicable, for reversing the effects of the operation (e.g., an HMAC key for keyed hashing, or an AES key for encryption and decryption). Also known as a secret-key algorithm.
Target data	The data that is to be protected (e.g., a key or other sensitive data).
Trusted Party	A party that is trusted by its clients to generate cryptographic keys.

2.2 Acronyms

AES	Advanced Encryption Standard. See FIPS 197.[7]

[7] FIPS 197, *Advanced Encryption Standard*.

CMAC	Cipher-based MAC. See SP 800-38B.[8]
CTR	Counter mode for a block cipher algorithm. See SP 800-38A.[9]
DSA	Digital Signature Algorithm. See FIPS 186.[10]
ECDSA	Elliptic Curve Digital Signature Algorithm. See FIPS 800-186.
FIPS	Federal Information Processing Standard.
HMAC	Keyed-Hash Message Authentication Code. See FIPS 198.[11]
KDF	Key Derivation Function.
KMAC	KECCAK Message Authentication Code. See SP 800-185.[12]
MAC	Message Authentication Code.
NIST	National Institute of Standards and Technology.
RBG	Random Bit Generator.
RSA	Rivest-Shamir-Adleman.
SP	Special Publication.

2.3 Symbols

Symbol	Meaning
$\oplus$	Bit-wise exclusive-or. A mathematical operation that is defined as: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, and $1 \oplus 1 = 0$.
$\|\|$	Concatenation
$H(x)$	A cryptographic hash function with x as an input.
$\min(x, y)$	The minimum of x and y; $\min(x, y) = x$ if $x <$ y, and $\min(x, y) = y$ otherwise.
$T(x, k)$	Truncation of the bit string x to the leftmost k bits of x, where $k \leq$ the length of x in bits.

[8] SP 800-38B, *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication.*
[9] SP 800-38A, *Recommendation for Block Cipher Modes of Operation - Methods and Techniques*.
[10] FIPS 186, *Digital Signature Algorithm (DSS).*
[11] FIPS 198, *Keyed-Hash Message Authentication Code (HMAC).*
[12] SP 800-185, *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash.*

3 General Discussion

3.1 Keys to Be Generated

This Recommendation addresses the generation of the cryptographic keys used in cryptography. Key generation includes the generation of a key using the output of a Random Bit Generator (RBG), the derivation of a key from another key, the derivation of a key from a password, and key agreement performed by two entities using an **approved** key-agreement scheme. All keys **shall** be based directly or indirectly on the output of an **approved** RBG. For the purposes of this Recommendation, keys that are derived during a key-agreement transaction (see SP 800-56A and SP 800-56B), derived from another key using a key derivation function (see SP 800-108) or derived from a password for storage applications (see SP 800-132[13] and Section 6.5) are considered to be indirectly generated from an RBG, since an ancestor key[14] or random value (e.g., the random value used to generate a key-agreement key pair) was obtained directly from the output of an **approved** RBG.

Two classes of cryptographic algorithms that require cryptographic keys have been **approved** for Federal government use: asymmetric-key algorithms and symmetric-key algorithms. The generation of keys for these algorithm classes is discussed in Sections 5 and 6.

3.2 Where Keys are Generated

Cryptographic keys **shall** be generated within FIPS 140-approved cryptographic modules. For explanatory purposes, consider the cryptographic module in which a key is generated to be the key-generating module. Any random value required by the key-generating module **shall** be generated within that module; that is, the RBG (or portion of the RBG[15]) that generates the random value **shall** be implemented within the FIPS 140 cryptographic module that generates the key. The generated keys **shall** be transported (when transportation is necessary) using secure channels and **shall** be used by their associated cryptographic algorithm within FIPS 140-**approved** cryptographic modules.

3.3 Supporting a Security Strength

A method (e.g., an RBG, or a key and its associated cryptographic algorithm) *supports a given security strength* if the security strength provided by that method is equal to or greater than the security strength required for protecting the target data; the actual security strength provided can be higher than required.

Security strength supported by an RBG: A well-designed RBG supports a given security strength if the amount of entropy (i.e., randomness) available in the RBG is equal to or greater than that security strength. The security strength supported depends on the secrecy

[13] SP 800-132, *Recommendation for Password-Based Key Derivation, Part 1: Storage Applications*.

[14] Ancestor key: A key that is used in the generation of another key. For example, an ancestor key for a key generated by a key derivation function would be the key-derivation key used by that key derivation function.

[15] The RBG itself might be distributed (e.g., the entropy source may not co-reside with the algorithm that generates the (pseudo) random output).

of the information designated as the entropy bits and, when used for the generation of keys and other secret values, on the secrecy of the RBG output. For information regarding the security strength that can be supported by **approved** RBGs, see SP 800-90A.[16]

Security strength supported by an algorithm: Discussions of cryptographic algorithms and the security strengths they can support, given certain choices of parameters and/or key lengths, are provided in SP 800-57, Part 1.[17] The security strength of a cryptographic algorithm that uses keys of a certain size (i.e., length) is assessed under the assumption that those keys are generated using an **approved** process that provides entropy equal to or greater than the security strength assessed for that algorithm and key size (where both the entropy and the security strength are measured in bits).

Security strength supported by a key: The security strength that can be supported by a key depends on 1) the algorithm with which it is used, 2) the size of the key (see SP 800-57, Part 1), 3) the process that generated the key (e.g., the security strength supported by the RBG that was used to generate the key), and 4) how the key was handled (e.g., the security strength available in the method used to transport the key). The use of such terms as "security strength supported by a key" or "key supports a security strength" assumes that these factors have been taken into consideration. For example, if an **approved** RBG that supports a security strength of 128 bits has been used to generate a 128-bit key, and if (immediately after generation) the key is used with AES-128 to encrypt target data, then the key may be said to support a security strength of 128 bits in that encryption operation (for as long as the key is kept secret). However, if the 128-bit AES key is generated using an RBG that supports a security strength of only 112 bits, then the key can only support a security strength of 112 bits, even though its length is still 128 bits; i.e., the security strength of the key has been reduced because of the process used for its generation (see item 3 above).

4 Using the Output of a Random Bit Generator

Random bit strings required for the generation of cryptographic keys **shall** be obtained from the output of an **approved** Random Bit Generator (RBG); **approved** RBGs are specified in SP 800-90.[18] The RBG **shall** be instantiated at a security strength that supports the security strength required to protect target data (i.e., the data that will be protected by the generated keys).

The output of an **approved** RBG may be used as specified in this section to obtain either a symmetric key or the random value needed to generate an asymmetric key pair.

Asymmetric key pairs require the use of an **approved** algorithm for their generation. Examples are those included in FIPS 186 for generating DSA, ECDSA and RSA keys. The generation of asymmetric key pairs from a random value is discussed in Section 5.

[16] SP 800-90A, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*.
[17] SP 800-57, Part 1: *Recommendation for Key Management: General*.
[18] SP 800-90, *Recommendation for Random Number Generation*, consisting of SP 800-90A, SP 800-90B and SP 800-90C.

Methods for the generation of symmetric keys are discussed in Section 6.

Let K be either a symmetric key or the random value to be used as input to an **approved** asymmetric-key pair generation algorithm. K **shall** be a bit string value of the following form:

$$K = U \oplus V, \tag{1}$$

where

- U is a bit string of the desired length that is obtained as the output of an **approved** RBG that is capable of supporting the desired security strength required to protect the target data,
- V is a bit string of the same length as U, and
- The value of V is determined in a manner that is independent of the value of U (and vice-versa).

The algorithm with which K will be used, and the security strength that this usage is intended to support will determine the required bit length and/or the security strength that this process must provide. Since there are no restrictions on the selection of V (other than its length and its independence from U), a conservative approach necessitates an assumption that the process used to select U provides most (if not all) of the required entropy.

The independence requirement on U and V is interpreted in a computational and a statistical sense; that is, the computation of U does not depend on V, the computation of V does not depend on U, and knowing one of the values (U or V) must yield no additional information that can be used to gain insight into the other value. Assuming that U is the output of an **approved** RBG, the following are examples of independently selected V values:

1. V is a constant (selected independently of the value of U). (Note, that if V is a string of binary zeroes, then $K = U$, i.e., the output of an **approved** RBG.)
2. V is a key obtained using an **approved** key-derivation method from a key-derivation key and other input that is independent of U; see SP 800-108.
3. V is a key that was independently generated in another cryptographic module. V was protected using an **approved** key-wrapping algorithm or transported using an **approved** key-transport scheme during subsequent transport. Upon receipt, the protection on V is removed within the key-generating module that generated U before combining V with U.
4. V is produced by hashing another bit string (V') using an **approved** hash function and (if necessary) truncating the result to the appropriate length before combining it with U. That is, $V = \mathrm{T}(\mathrm{H}(V'), k)$ where $\mathrm{T}(x, k)$ denotes the truncation of bit string x to its k leftmost bits, and k is the length of U. The bit string V' may be a) a constant; b) a key derived from a shared secret during an **approved** key-agreement scheme between the key-generating module and another cryptographic module; or c) a key that was i) independently generated by another module, ii) sent using an

approved key wrapping algorithm or transported using an **approved** key transport scheme, and iii) upon receipt, the protection on the key was removed.

5 Generation of Key Pairs for Asymmetric-Key Algorithms

Asymmetric-key algorithms, also known as public-key algorithms, require the use of asymmetric key pairs, consisting of a private key and a corresponding public key. A key pair can be used for the generation and verification of digital signatures (see Section 5.1) or for key establishment (see Section 5.2). Each public/private key pair is associated with only one entity; this entity is known as the key-pair owner. Key pairs **shall** be generated by:

- The key-pair owner, or
- A Trusted Party that provides the key pair to the owner in a secure manner. The Trusted Party must be trusted by all parties that use the public key.

After key-pair generation, the key pair is retained and used by its owner; if the key pair was generated by a Trusted Party, both the owner and any relying party must trust that party not to use the private key of the key pair. The public key may be known by and distributed to whomever needs to use it when interacting with the owner (see Section 5.3).

5.1 Key Pairs for Digital Signature Schemes

Digital signatures are generated on data to provide origin authentication, entity authentication, assurance of data integrity or support for signatory non-repudiation. Digital signatures are generated by a signer using a private key and verified by a receiver using a public key. The generation of key pairs for digital signature applications is addressed in FIPS 186 for the DSA, RSA and ECDSA digital signature algorithms.

A value of *K*, computed as shown in Section 4**, shall** be used to provide the random value needed to generate a key pair for the algorithms as specified in FIPS 186. The maximum security strength that can be supported by these algorithms and the key lengths used by these algorithms are provided in SP 800-57, Part 1.

For example, SP 800-57, Part 1 states that an ECDSA key pair with a key size of 224-255 bits can support (at most) a security strength of 112 bits. FIPS 186 specifies that for such DSA key pairs, the random value used to determine a private key must be obtained using an RBG that supports a security strength of 112 bits. Using the method in Section 4, a random value *K* that is to be used for the generation of the key pair is determined by *U* (a value of an appropriate bit length obtained from an RBG that supports a security strength of at least 112 bits) and *V* (which could be zero). The value *K* is then used to determine the ECDSA key pair, as specified in FIPS 186.

5.2 Key Pairs for Key Establishment

Key establishment includes both key agreement and key transport. Key agreement is a method of key establishment in which the resultant secret keying material is a function of information contributed by all participants in the key-establishment process (usually only two participants) so that no party can predetermine the value of the keying material

independent of any other party's contribution. For key-transport, one party (the sender) selects a value for the secret keying material and then securely distributes that value to one or more other parties (the receiver).

Approved methods for generating the (asymmetric) key pairs used by **approved** key-establishment schemes between two parties are specified in SP 800-56A (for schemes that use finite-field or elliptic-curve cryptography) and in SP 800-56B (for schemes that use integer-factorization cryptography, e.g., RSA).

Values of *K*, computed as shown in Section 4, **shall** be used to provide the random values[19] needed to generate key pairs for the finite field or elliptic curve schemes in SP 800-56A, or to generate key pairs for the integer-factorization schemes specified in SP 800-56B. The maximum security strength that can be supported by the **approved** key-establishment schemes and the key sizes used by these schemes is provided in SP 800-57, Part 1.

5.3 Distributing the Key Pairs

A general discussion of the distribution of asymmetric key pairs is provided in SP 800-57, Part 1. Key pairs may be either static or ephemeral. Static key pairs are intended to be used multiple times; ephemeral keys are usually used only once.

The private key of a key pair **shall** be kept secret. It **shall** either be generated 1) within the key-pair owner's cryptographic module (i.e., the key-pair owner's key-generating module), or 2) within the cryptographic module of an entity trusted by the key-pair owner and any relying party not to misuse the private key or reveal it to other entities (i.e., generated within the key-generating module of a Trusted Party and securely transferred to the key-pair owner's cryptographic module).

If a private key is ever output from a cryptographic module, the key **shall** be output and transferred in a form and manner that provides appropriate assurance[20] of its confidentiality and integrity (e.g., using manual methods and multiparty control procedures or using automated key-transport methods). The protection **shall** provide appropriate assurance that only the key-pair owner and/or the party that generated the key pair will be able to determine the value of the plaintext private key (e.g., the confidentiality and integrity protection for the private key uses a cryptographic mechanism that is at least as strong as the (maximum) security strength that must be supported by the asymmetric algorithm that will use the private key).

The public key of a key pair may be made public. However, it **shall** be distributed and verified in a manner that assures its integrity and association with the key-pair owner (e.g., in the case of a static public key, this may be accomplished using an X.509 certificate that provides a level of cryptographic protection that is at least as strong as the security strength associated with the key pair).

[19] Note that in Section 4, if *V* is all zeroes, then *K* (the random value) is the output of an RBG.

[20] The term "provide appropriate assurance" is used to allow various methods for the input and output of critical security parameters to/from the different security levels that may be implemented for a cryptographic module (see FIPS 140 and Section 7.8 of the FIPS 140 IG).

5.4 Key Pair Replacement

Key pairs need to be replaced if the private key is compromised. Key pairs also need to be replaced from time to time to limit the amount of information that is protected by the key pair in case of a compromise of the private key (see Section 5.3 of SP 800-57 Part 1); Section 5.3.4 of SP 800-57 Part 1 discusses the usage period for each key of the key pair for both digital signature and key-establishment key pairs.

When asymmetric key pairs need to be replaced, they **shall** be generated and distributed as specified in Sections 5.1, 5.2 or 5.3, as appropriate.

6 Generation of Keys for Symmetric-Key Algorithms

Symmetric-key algorithms use the same (secret) key to both apply cryptographic protection to information[21] and to remove or verify the protection.[22] Keys used with symmetric-key algorithms must be known by only the entities authorized to apply, remove or verify the protection, and are commonly known as secret keys. A secret key is often known by multiple entities that are said to share or own the secret key, although it is not uncommon for a key to be generated, owned and used by a single entity (e.g., for secure storage). A secret key **shall** be generated by:

- One or more of the entities that will share the key, or
- A Trusted Party that provides the key to the intended sharing entities in a secure manner. The Trusted Party must be trusted by all entities that will share the key not to disclose the key to unauthorized parties or otherwise misuse the key (see SP 800-71[23]).

A symmetric key *K* could be used, for example, to:

- Encrypt and decrypt data in an appropriate mode (e.g., using AES in the CTR mode as specified in FIPS 197 and SP 800-38A),
- Generate Message Authentication Codes (e.g., using AES in the CMAC mode, as specified in FIPS 197 and SP 800-38B; HMAC, as specified in FIPS 198; or KMAC, as specified in SP 800-185.
- Derive additional keys using a key-derivation function specified in SP 800-108, where *K* is the pre-shared key that is used as the key-derivation key.

Section 6.1 discusses the generation of keys that are generated from the output of an RBG. Section 6.2 discusses the distribution of a symmetric key after generation to the parties that need to use it. Section 6.3 discusses a key agreement process using symmetric keys, whereby keys are generated using a method that does not require explicit distribution of

[21] For example, transform plaintext data into ciphertext data using an encryption operation or compute a message authentication code (MAC).

[22] For example, remove the protection by transforming the ciphertext data back to the original plaintext data using a decryption operation, or verify the protection by computing a message authentication code and comparing the newly computed MAC with a received MAC)

[23] SP 800-71, *Recommendation for Key Establishment Using Symmetric Block Ciphers*.

the generated key, since the key is already made available to the parties that need to use it during the key-generation process. Once two or more parties share an appropriate key, additional keys may be derived from that key without explicitly distributing those keys (see Section 6.4).

Sometimes symmetric keys are not intended to be shared, e.g., those keys intended to encrypt the key owner's data in storage. The preferred method for key generation is as discussed in Section 6.1, but without distributing the key to other parties. Another method is to use a password known only by the data's owner to generate a storage key; however, this method is less secure than the previous methods (see Section 6.5).

A symmetric key can be combined with other keys or data to create another symmetric key to be used for cryptographic protection (see Section 6.6).

At some point, a symmetric key needs to be replaced for a number of possible reasons, e.g., its cryptoperiod has been exceeded or it has been compromised (see SP 800-57 Part 1); Section 6.7 discusses key replacement.

6.1 The "Direct Generation" of Symmetric Keys

Symmetric keys that are to be directly generated from the output of an RBG **shall** be generated as specified in Section 4, where *K* is the desired key. The length of the key to be generated is determined by the algorithm with which it will be used and the desired security strength to be provided; see SP 800-57, Part 1 for discussions on key lengths and the (maximum) security strengths supported by symmetric-key algorithms.

6.2 Distributing the Generated Symmetric Key

The symmetric key generated within a key-generating module often needs to be shared with one or more other entities that have their own cryptographic modules. The key may be distributed manually or using an **approved** key-transport or key-wrapping method (see SP 800-56B, SP 800-38F and SP 800-71). See SP 800-57, Part 1 for further discussion. The method used for key transport or key wrapping **shall** support the desired security strength needed to protect the target data (i.e., the data to be protected using the symmetric key). The requirements for outputting a key from a cryptographic module are discussed in FIPS 140.

6.3 Symmetric Keys Generated Using Key-Agreement Schemes

When an **approved** key-agreement scheme is available within an entity's key-generating module, a symmetric key may be established with another entity that has the same capability; this process results in a symmetric key that is shared between the two entities participating in the key-agreement transaction; further distribution of this symmetric key is not required between the two entities.

SP 800-56A and SP 800-56B provide several methods for pairwise key agreement. Asymmetric key-agreement keys are used with a key-agreement primitive algorithm to generate a shared secret. The shared secret is provided to a key-derivation method to derive

keying material. SP 800-56C[24] specifies **approved** key-derivation methods for the key-agreement schemes in SP 800-56A and SP 800-56B.

The maximum security strength that can be supported by a key derived in this manner is dependent on 1) the security strength supported by the asymmetric key pairs (as used during key establishment), 2) the key-derivation method used, 3) the length of the derived key and 4) the algorithm with which the derived key will be used. See SP 800-57, Part 1.

6.4 Symmetric Keys Derived from a Pre-shared Key

Symmetric keys are often derived using a key-derivation function (KDF) and a pre-shared key known as a key-derivation key. The pre-shared key may have been:

- Generated from an **approved** RBG (see Section 4) and distributed as specified in Section 6.2, if required,
- Agreed upon using a key-agreement scheme (see Section 6.3), or
- Derived using a KDF and a (different) pre-shared key as specified in SP 800-108.

Approved methods for key derivation are provided in SP 800-108, which specifies **approved** KDFs for deriving keys from a pre-shared key (i.e., a key-derivation key). The KDFs are based on HMAC (as specified in FIPS 198) and CMAC (as specified in SP 800-38B).

If the derived keys need to be distributed to other entities, this may be accomplished as discussed in Section 6.2.

In addition to the symmetric-key algorithm with which a derived key will be used, the security strength that can be supported by the derived key depends on the security strength supported by the key-derivation key and the KDF method used (see SP 800-57, Part 1 for the maximum security strength that can be supported by HMAC and CMAC, and SP 800-107[25] for further discussions about the security strength of HMAC).

6.5 Symmetric Keys Derived from Passwords

In a number of popular applications, keys are generated from passwords. This is a questionable practice, as the passwords usually contain very little entropy (i.e., randomness), and are, therefore, easily guessed. However, **approved** methods for deriving keys from passwords for storage applications[26] are provided in SP 800-132. For these applications, users are strongly advised to select passwords with a very large amount of entropy.

When a key is generated from a password, the entropy provided (and thus, the maximum security strength that can be supported by the generated key) **shall** be considered to be zero unless the password is generated using an **approved** RBG. In this case, the security

[24] SP 800-56C, *Recommendation for Key-Derivation Methods in Key-Establishment Schemes*.

[25] SP 800-107, *Recommendation for Applications Using Approved Hash Algorithms*.

[26] For example, inside a FIPS 140-validated cryptographic module.

strength that can be supported by the password (*password_strength*) is no greater than the minimum of the security strength supported by the RBG (*RBG_strength*) and the actual number of bits of RBG output (*RBG_outlen*) used in the password. That is, *password_strength* ≤ min(*RBG_strength*, *RBG_outlen*).

6.6 Symmetric Keys Produced by Combining Multiple Keys and Other Data

When symmetric keys K_1, …, K_n are generated and/or established independently, they may be combined within a key-generating module to form a key K. Other items of data (V_1, …, V_m) that are independent of these keys can also be combined with the K_i to form K, under the conditions specified below. Note that, while the K_i values are required to be secret, the V_i values need not be kept secret.

The independent generation/establishment of the component keys K_1, …, K_n is interpreted in a computational and a statistical sense; that is, the computation of any particular K_i value does not depend on any one or more of the other K_i values, and it is not feasible to use knowledge of any proper subset of the K_i values to obtain any information about the remaining K_i values.

The independence of the component keys from the other items of data is also interpreted in a computational and a statistical sense. This means that the computation of the K_i values does not depend on any of the V_j values, the computation of the V_j values does not depend on any of the K_i values, and knowledge of the V_j values yields no information that can feasibly be used to gain insight into the K_i values. In cases where some (or all) of the V_j values are secret, and the rest of the V_j values (if any) are public, "independence" also means that knowledge of the K_i values and public V_j values yields no information that can feasibly be used to gain insight into the secret V_j values.

The component symmetric keys K_1, …, K_n **shall** be generated and/or established independently using **approved** methods.[27] The other items of data (i.e., V_1, …, V_m) may be generated or obtained using any method that ensures their independence from those keys.

The **approved** methods for combining these symmetric keys (and other items of data) are:

1. Concatenating two or more keys, i.e., $K = K_1 \| \ldots \| K_n$. The sum of the bit lengths of the n component keys **shall** be equal to the required bit length of K.
2. Exclusive-Oring one or more symmetric keys, i.e., $K = K_1 \oplus \ldots \oplus K_n$. The length of each component key (K_i) **shall** be equal to the length required for K.
3. Exclusive-Oring one or more symmetric keys and one or more other items of data, i.e., $K = K_1 \oplus \ldots \oplus K_n \oplus V_1 \oplus \ldots \oplus V_m$. The length of each component key (K_i) and each item of data (V_i) **shall** be equal to the length required for K.

Each K_i used in an **approved** method for combing symmetric keys **shall** be kept secret and **shall not** be used for any purpose other than the computation of a specific symmetric key K (i.e., a given K_i **shall not** be used to generate more than one key).

[27] See Sections 6.1, 6.3 and 6.4.

6.7 Replacement of Symmetric Keys

Sometimes, a symmetric key may need to be replaced. This may be due to a compromise of the key or the end of the key's cryptoperiod (see SP 800-57, Part 1). Replacement **shall** be accomplished by a rekeying process. Rekeying is the replacement of a key with a new key that is generated independent of the value of the old key (i.e., knowledge of the old key provides no knowledge of the value of the replaced key and vice versa).

When a compromised key is replaced, the new key **shall** be generated in a manner that provides assurance of its independence from the compromised key. The new key may be generated using any method in Section 5 with the following restrictions:

1. The method used **shall** provide assurance that there is no feasibly detectable relationship between the new key and the compromised key. To that end, the new key **shall not** be derived or updated using the compromised key.
2. If the compromised key was generated in a manner that depended (in whole or in part) on a password (see Sections 6.5 and 6.6), then that password **shall** be changed prior to the generation of any new key; in particular, the new key(s) **shall** be generated in a manner that is independent of the old password value.

If an uncompromised symmetric key is to be replaced, it **shall** be replaced using any method in Section 6 that supports the required amount of security strength. However, if the key to be replaced was generated in a manner that depended (in whole or in part) on a password (see Sections 6.5 and 6.6), that password **shall** be changed prior to the generation of the new key.

Appendix A: References

[FIPS 140] National Institute of Standards and Technology (2019) *Security Requirements for Cryptographic Modules*. (U.S. Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) 140-3. https://doi.org/10.6028/NIST.FIPS.140-3

[FIPS 140 IG] National Institute of Stanards and Technology, Canadian Centre for Cyber Security (2003) *Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*, [Amended]. Available at https://csrc.nist.gov/csrc/media/projects/cryptographic-module-validation-program/documents/fips140-2/FIPS1402IG.pdf

[FIPS 180] National Institute of Standards and Technology (2015) *Secure Hash Standard (SHS)*. (U.S. Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) 180-4. https://doi.org/10.6028/NIST.FIPS.180-4

[FIPS 186-4] National Institute of Standards and Technology (2013) *Digital Signature Standard (DSS)*. (U.S. Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) 186-4. https://doi.org/10.6028/NIST.FIPS.186-4

[FIPS 197] National Institute of Standards and Technology (2001) *Advanced Encryption Standard (AES)*. (U.S. Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) 197. https://doi.org/10.6028/NIST.FIPS.197

[FIPS 198] National Institute of Standards and Technology (2008) *The Keyed-Hash Message Authentication Code (HMAC)*. (U.S. Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS) 198-1. https://doi.org/10.6028/NIST.FIPS.198-1

[SP 800-38A] Dworkin MJ (2001) *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-38A. https://doi.org/10.6028/NIST.SP.800-38A

[SP 800-38B] Dworkin MJ (2016) *Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-38B, Includes updates as of October 06, 2016. https://doi.org/10.6028/NIST.SP.800-38B

[SP 800-38F] Dworkin MJ (2012) *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-38F. https://doi.org/10.6028/NIST.SP.800-38F

[SP 800-56A] Barker EB, Chen L, Roginsky A, Vassilev A, Davis R (2018) *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-56A, Rev. 3. https://doi.org/10.6028/NIST.SP.800-56Ar3

[SP 800-56B] Barker EB, Chen L, Roginsky A, Vassilev A, Davis R, Simon S (2019) *Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-56B, Rev. 2. https://doi.org/10.6028/NIST.SP.800-56Br2.

[SP 800-56C] Barker EB, Chen L, Davis R (2018) *Recommendation for Key-Derivation Methods in Key-Establishment Schemes*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-56C, Rev. 1. https://doi.org/10.6028/NIST.SP.800-56Cr1

[SP 800-57-1] Barker EB (2016) *Recommendation for Key Management, Part 1: General*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-57 Part 1, Rev. 4. https://doi.org/10.6028/NIST.SP.800-57pt1r4

[SP 800-67] Barker EB, Mouha N (2017) *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-67, Rev. 2. https://doi.org/10.6028/NIST.SP.800-67r2

[SP 800-71] Barker EB, Barker WC (2018) *Recommendation for Key Establishment Using Symmetric Block Ciphers.* (National Institute of Standards and Technology, Gaithersburg, Maryland), Draft NIST Special Publication (SP) 800-71. Available at https://csrc.nist.gov/publications/detail/sp/800-71/draft

[SP 800-90A] Barker EB, Kelsey JM (2015) *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-90A, Rev. 1. https://doi.org/10.6028/NIST.SP.800-90Ar1

[SP 800-90B] Sönmez Turan M, Barker EB, Kelsey JM, McKay KA, Baish ML, Boyle M (2018) *Recommendation for the Entropy Sources Used for Random Bit Generation*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-90B. https://doi.org/10.6028/NIST.SP.800-90B

[SP 800-90C] Barker EB, Kelsey JM (2016) *Recommendation for Random Bit Generator (RBG) Constructions*. (National Institute of Standards and

Technology, Gaithersburg, Maryland), Draft NIST Special Publication (SP) 800-90C. Available at https://csrc.nist.gov/publications/detail/sp/800-90c/draft

[SP 800-107] Dang QH (2012) *Recommendation for Applications Using Approved Hash Algorithms*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-107, Rev. 1. https://doi.org/10.6028/NIST.SP.800-107r1

[SP 800-108] Chen L (2009) *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-108. https://doi.org/10.6028/NIST.SP.800-108

[SP 800-131A] Barker EB, Roginsky A (2019) *Transitioning the Use of Cryptographic Algorithms and Key Lengths*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-131A, Rev. 2. https://doi.org/10.6028/NIST.SP.800-131Ar2.

[SP 800-132] Sönmez Turan M, Barker EB, Burr WE, Chen L (2010) *Recommendation for Password-Based Key Derivation, Part 1: Storage Applications*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-132. https://doi.org/10.6028/NIST.SP.800-132

[SP 800-135] Dang QH (2011) *Recommendation for Existing Application-Specific Key Derivation Functions*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-135, Rev. 1. https://doi.org/10.6028/NIST.SP.800-135r1

[SP 800-185] Kelsey, J, Chang, S., Perlner, R (2016) *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash*. (National Institute of Standards and Technology, Gaithersburg, Maryland), NIST Special Publication (SP) 800-185. https://doi.org/10.6028/NIST.SP.800-185.

Appendix B: Revisions

A revision was made in 2019 with the following changes:

1. General: The Authority section (old Section 2) has been moved into the boilerplate (see page iii). This resulted in a renumbering of the sections in the document.
2. Footnotes have been added to define each document when first mentioned.
3. Section 2.1: Changes made to *cryptographic boundary*, *entropy*, *key-pair owner*, *key transport*, *key wrapping, rekey, shared secret* and *target data.*

   Added: *entity authentication*, *KMAC*.

   Removed *full-entropy*, *key update* and *non-repudiation*.

4. Section 2.2: Added *KMAC* and *DSA*.

   Removed *DLC* and *IFC*.
5. Section 3.3, para. 2, last line: Changed the reference to SP 800-90A instead of SP 800-90.

   Last para.: The example has been expanded.
6. Section 4, para. 1, line 3: Removed the references to FIPS 186-2, X9.31 and X9.62, since the use of these RBGs is no longer allowed (see SP 800-131A).
7. Section 5: Rewrote the text and inserted guidance on handling a key pair after generation.
8. Section 5.1, para. 1, lines 1-2: Inserted entity authentication.
9. Section 5.3, Rewrote the text. Para. 3, lines 3-4: Inserted a parenthetical example.
10. Section 5.4: Added a new section on key replacement.
11. Section 6, bullet 2: Inserted a reference to SP 800-71. Bullet 4: Added KMAC, as specified in SP 800-185. Also added text introducing the remainder of Section 6.
12. Section 6.2, line 4: Inserted a reference to SP 800-71 and removed a reference to SP 800-56A.

123. Section 6.3: Removed the figure and some of the associated text. Last paragraph: Removed the last four lines.

14. Section 6.6: enlarged the subscripts for easier reading.
15. Section 6.7: The first paragraph was rewritten.
16. Appendix A: Updated the references.