Withdrawn Draft

Warning Notice

The attached draft document has been withdrawn, and is provided solely for historical purposes. It has been superseded by the document identified below.

Withdrawal Date	June 4, 2020
Original Release Date	March 5, 2020

Superseding Document

Status	Final
Series/Number	NIST Special Publication 800-133 Revision 2
Title	Recommendation for Cryptographic Key Generation
Publication Date	June 2020
DOI	https://doi.org/10.6028/NIST.SP.800-133r2
CSRC URL	https://csrc.nist.gov/publications/detail/sp/800-133rev-2/final
Additional Information	Key Management https://csrc.nist.gov/projects/key-management

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

Draft NIST Special Publication 800-133
Revision 2

Recommendation for Cryptographic Key Generation

Elaine Barker
Allen Roginsky
Richard Davis

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-133r2-draft

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

Draft NIST Special Publication 800-133 Revision 2

Recommendation for Cryptographic Key Generation

Elaine Barker
Allen Roginsky
Computer Security Division
Information Technology Laboratory

Richard Davis
National Security Agency

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-133r2-draft

March 2020

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 2
Natl. Inst. Stand. Technol. Spec. Publ. 800-133 Rev 2, 36 pages (February 2020)
CODEN: NSPUE2

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-133r2-draft

Public comment period: *March 5, 2020* through *April 17, 2020*

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.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or
   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: SP-800-133_Comments@nist.gov

Table of Contents

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, storage, or use. Cryptography relies upon two basic components: an algorithm (or cryptographic methodology) and, often, 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 use by the Federal Government. In addition, guidance has been provided on the management of the cryptographic keys to be used with these **approved** cryptographic algorithms.

This Recommendation (i.e., SP 800-133) discusses the generation of the keys to be used with the **approved** cryptographic algorithms. The keys are either 1) generated using mathematical processing on the output of **approved** Random Bit Generators (RBGs) and possibly other parameters or 2) generated based on 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. [1]

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

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 a message authentication code (MAC) from data, 6. The verification of a MAC from data and a received MAC, 7. The computation of a shared secret that is used to derive keying material, and 8. The derivation of additional keying material from a key-derivation key (e.g., a pre-shared key).
Cryptographic module	The set of hardware, software, and/or firmware that implements security functions (including cryptographic algorithms and key-generation methods) and is contained within a cryptographic module boundary. See FIPS 140.
Cryptoperiod	The timespan 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.
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 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 function	As used in this Recommendation, either a one-step key-derivation method or a key-derivation function based on a pseudorandom function as specified in SP 800-108.
Key-derivation key	A key used as an input to a key-derivation method to derive other keys; see SP 800-108.[5]
Key-derivation method	A key-derivation function or other **approved** procedure for deriving keying material.
Key-derivation procedure	As used in this Recommendation, a two-step key-derivation method consisting of randomness extraction followed by key expansion.
Key establishment	A procedure that results in secret keying material that is shared among different parties.

[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 expansion	The second step in the key-derivation procedure in which a key-derivation key is used to derive secret keying material having the desired length(s). The first step in the procedure is randomness extraction.
Key extraction	See Randomness extraction.
Key-generating module	A cryptographic module in which a given key is generated.
Key generation	The process of generating keys for cryptography.
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]
Key-wrapping key	A key used as an input to a key-wrapping method; see SP 800-38F.
MAC key	A symmetric key used as input to a security function to produce a message authentication code (MAC).
Message Authentication Code (MAC)	A cryptographic checksum on data that uses an **approved** security function and a symmetric key to detect both accidental and intentional modifications of data.
Min-entropy	The min-entropy (in bits) of a random variable *X* is the largest value *m* having the property that each observation of *X* provides at least *m* bits of information (i.e., the min-entropy of *X* is the greatest lower bound for the information content of potential observations of *X*). The min-entropy of a random variable is a lower bound on its entropy. The precise formulation for min-entropy is $-(\log_2(\max p_i))$ for a discrete distribution having probabilities $p_1, \ldots, p_k$. Min-entropy is often used as a worst-case measure of the unpredictability of a random variable.

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

Module	See Cryptographic module.
Nonce	A time-varying value that has (at most) an acceptably small chance of repeating. For example, the nonce may be a random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these.
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 the 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 Recommendation, 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.
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).
Primitive algorithm	A low-level cryptographic algorithm (e.g., an RSA encryption operation) used as a basic building block for higher-level cryptographic algorithms or schemes (e.g., RSA key transport).

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.
Randomness extraction	The first step in the two-step key-derivation procedure during which a key-derivation key is produced. The second step in the procedure is key expansion.
Recommendation	A term used to refer to this specific document (i.e., SP 800-133): the “R” is always capitalized.
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.

Salt	As used in this Recommendation, a byte string (which may be secret or non-secret) that is used as a MAC key.
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 function	Cryptographic algorithms, together with modes of operation (if appropriate); for example, block ciphers, digital signature algorithms, asymmetric key-establishment algorithms, message authentication codes, hash functions, or random bit generators; see FIPS 140.
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. A security strength of s bits is said to be supported by a particular choice of keying material, algorithm, primitive, auxiliary function, parameters (etc.) for use in the implementation of a cryptographic mechanism if that choice will not prevent the resulting implementation from attaining a security strength of at least s bits.
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]
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 186
FIPS	Federal Information Processing Standard
HMAC	Keyed-Hash Message Authentication Code; see FIPS 198[11]
KDF	Key-Derivation Function
KDM	Key-Deviation Method
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

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

2.3 Symbols and Terms

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
B	The bit string to be determined
$bLen$	The length of the bit string B in bits
$\mathrm{H}(x)$	A cryptographic hash function with x as an input
K	The key to be determined
$kLen$	The length of K in bits
$\max(x_1, \ldots, x_n)$	The maximum of the x_i values
$\min(x, y)$	The minimum of x and y; $\min(x, y) = x$ if $x <$ y, and $\min(x, y) = y$ otherwise
ss_K	The security strength that can be supported by the key K
ss_M_i	The security strength that can be supported by the combination of the methods used to generate a key K_i, and the methods used to protect it after generation (e.g., during key-transport and/or storage)
$\mathrm{T}(x, l)$	Truncation of the bit string x to the leftmost l bits of x, where $l \leq$ the length of x in bits

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 use by the Federal Government: asymmetric-key algorithms and symmetric-key algorithms. The generation of keys for these algorithm classes is discussed in Sections 5 and 6, respectively.

3.2 Where Keys are Generated

Cryptographic keys **shall** be generated within FIPS 140-validated 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-validated 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 only if the amount of entropy (i.e., the randomness) available in the RBG is equal to or greater than that security strength. The support of a given security strength also requires a commensurate security strength for the confidentiality protection afforded to the entropy bits entered into the RBG (and other parameters determining the RBG's state); when used for the generation of keys and other

[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).

secret values, a commensurate security strength is also required for the confidentiality and integrity protection that will be provided to 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., the key's length) is assessed under the assumption that those keys are generated using an **approved** process that outputs keys of the requisite size and type while providing an appropriate amount of min-entropy. It is assumed that this key-generation process is capable of supporting security strengths that are equal to or greater than the security strength assessed for the cryptographic algorithm. (Both the min-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 determined by the process used for its generation).

[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*.

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 the target data (i.e., the data that will be protected by the generated keys).

The output of an **approved** RBG can be used as specified in this section to obtain, for example, 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.

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

When random bit strings are required for the generation of cryptographic keys, they are obtained as follows:

Let B be the random bit string to be acquired, for example, to use as a symmetric key (K), as input to an asymmetric-key-generation algorithm, or as part of the seed material for an RBG. Let $bLen$ be its desired length in bits. B **shall** be a bit string that is formed as follows:

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

where

- U is a bit string of $bLen$ bits that is obtained as the output of an **approved** RBG that is capable of supporting the security strength required by the algorithm and/or application using B (e.g., to protect the target data),
- V is a bit string of *bLen bits*, and
- The value of V is determined in a manner that is independent of the value of U (and vice versa).

The algorithm and/or application with which B will be used and the security strength that B is intended to support will determine the required bit length of B: $bLen$ **shall** meet the relying application or algorithm's length requirement for a value of the bit string B to be used as intended in support of the targeted security strength. For example, if B is to be used as an AES key, then $bLen$ **shall** be an **approved** AES key length that supports the required security strength for protecting the target data. As another example, according to FIPS 186, if B is to be used as a seed in the specified process of generating provably prime factors of an RSA modulus n, then $bLen$ **shall** be twice the security strength associated with (the bit length of) n.

Since there are no restrictions on the selection of V (other than its length and 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, which – when measured in bits – cannot exceed the length of U (i.e., $bLen$). Therefore, the **approved** RBG from which U is obtained **shall** be capable of

[18] SP 800-90, *Recommendation for Random Number Generation*, consisting of SP 800-90A, SP 800-90B, and SP 800-90C.

providing the requisite entropy for B during the generation of U (i.e., at least *bLen* bits of entropy are provided during the seeding of the RBG).

The independence requirement on U and V is interpreted in a computational and statistical sense: the computation of U does not depend on V, and the computation of V does not depend on U. Knowledge of the value of V (but not B) must provide no advantage to a party intent on gaining insight into an (as-yet-unknown) value of U. The value of V may be selected using a process that provides little entropy (indeed, V may be assigned a fixed, public value). Nevertheless, in cases where the value of V is intended to be kept secret, knowledge of the value of U (but not B) must yield no additional information concerning the value selected for V. Given 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). The value of V may be dependent on the use of B (e.g., if B is used as a key-derivation key, then V may be some value M, but if B is used as a key-wrapping key, then V may be some value N). Note that if V is a string of binary zeroes, then $B = U$ (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, $V = \mathrm{T}(\mathrm{H}(V'), bLen)$ where $\mathrm{T}(x, bLen)$ denotes the truncation of bit string x to its *bLen* leftmost bits. The bit string V' may be selected using methods 1, 2, or 3 above.

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 or provided 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.

Values of B, computed as shown in Section 4, **shall** be used to provide the random bit strings used in key-pair generation, as specified in FIPS 186. The maximum security strength that can be supported by the resulting key pairs depends on a variety of size and parameter choices. Guidance on the size/parameter choices appropriate for supporting various security strengths can be found in SP 800-57, Part 1.

For example, SP 800-57, Part 1 states that an ECDSA key pair generated using an appropriate elliptic curve and a base point whose order is a 224-bit to 255-bit prime number can support (at most) a security strength of 112 bits. FIPS 186 specifies that for such ECDSA 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 B that is to be used for the generation of the private key is determined by U (a value of a specified bit length obtained from an RBG that supports a security strength of at least 112 bits) and V (which could be zero). The value of B is then used to determine the private key from which the public ECDSA key is obtained, 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(s)).

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 SP 800-56B (for schemes that use integer-factorization cryptography, such as RSA).

Values of B, 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 either be 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., the key pair is 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 multi-party control procedures or 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-key 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).

5.4 Key Pair Replacement

Key pairs need to be replaced if the private key is compromised. Key pairs also need to be replaced occasionally 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,

[19] Note that in Section 4, if V is all zeroes, then B (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 cryptographic keys to/from cryptographic modules that may be implemented at different security levels (see FIPS 140 and Section 7.7 of the FIPS 140 IG).

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), or
- Derive additional keys using a key-derivation function specified in SP 800-108, where *K* is the pre-shared (i.e., pre-existing) key that is used as the key-derivation key (e.g., *K* could be a value of *B* generated as specified in Section 4).

Section 6.1 discusses the generation of symmetric keys that are obtained from the output of an RBG. Section 6.2 discusses the derivation of symmetric keys. Section 6.3 specifies **approved** techniques for combining a symmetric key with other symmetric keys and/or additional data.

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 6.4 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 *B* is used as the desired key *K*. The length of the key to be generated depends on the length requirement of the application or algorithm with which the key is used and the security strength to be supported. See SP 800-57, Part 1 for discussions on key lengths and the (maximum) security strengths supported by symmetric-key algorithms and their keys.

---

[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*.

6.2 Derivation of Symmetric Keys

Symmetric keys are often obtained from the output of an **approved** key-derivation method (KDM), which is a cryptographic process specifically designed to transform secret input values into bit strings that can be parsed into cryptographic keys and/or other secret keying material.

Approved KDMs have been constructed from more basic cryptographic components, such as an **approved** hash function, as specified in FIPS 180 or FIPS 202; HMAC (using an **approved** hash function), as specified in FIPS 198; AES-CMAC, as specified in FIPS 197 and SP 800-38B; or a KMAC variant, as specified in SP 800-185.

Depending on the application and the KDM, the input to a KDM may include, for example, one or more of the following:

- A shared secret value produced during the execution of a key-agreement scheme;
- A cryptographic key (i.e., a key-derivation key (KDK));
- A password or passphrase;
- A salt value, which may be secret or non-secret, fixed, or randomly selected;
- A nonce (including RBG output) that may, for example, indicate the algorithm to be associated with the key (e.g., AES), the use of the key (e.g., email), or any other information that may be useful for associating a particular execution of the KDM with the key(s) to be derived.

Approved key-derivation methods can be divided into two categories:

1) The first category consists of one-step key-derivation methods, which are usually called key-derivation functions (KDFs). General-purpose KDFs are based on pseudorandom functions (PRFs) that use a KDK (and other input) to generate additional keys (see SP 800-108). Some special-purpose KDFs, which are employed only as components of key-agreement schemes, are used to obtain keying material from the shared secrets produced during the execution of such schemes (see SP 800-56C and SP 800-135); other special-purpose KDFs are to be used only for password-based protection of stored data and/or the keys that protect that data (see SP 800-132).

2) The second category consists of extraction-then-expansion key-derivation procedures that involve two steps:

    a. Randomness extraction to obtain a single cryptographic key-derivation key. The extraction of a KDK from a shared secret produced during the execution of a key-agreement scheme is described in SP 800-56C. The HMAC-based extraction of a symmetric key from the concatenation of pre-existing symmetric keys (and, perhaps, other data) is described in Section 6.3 (along with other methods of combining preexisting keys to form a new key). The key resulting from a key-extraction process can be used as a KDK for key expansion.

b. Key expansion to derive keying material from 1) the key-derivation key produced during randomness extraction and 2) other information, as specified in SP 800-56C[24] and SP 800-108.

6.2.1 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.

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 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.

6.2.2 Symmetric Keys Derived from a Pre-existing Key

Symmetric keys are often derived using a key-derivation function (KDF) and a preexisting key known as a key-derivation key. For example, the preexisting key may have been:

- Generated from an **approved** RBG (see Section 4) and distributed as specified in Section 6.4;
- Agreed upon using a key-agreement scheme (see Section 6.2.1);
- Derived using a KDF and a (different) preexisting key as specified in SP 800-108; or
- The concatenation of multiple cryptographic keys (and, perhaps, other data) as described in Section 6.3.

Approved methods for key derivation are provided in SP 800-108, which specifies **approved** KDFs for deriving keys from a pre-shared (i.e., preexisting) 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 6.4.

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 used (see SP 800-57, Part 1 for the maximum security strength

[24] When the two-step key-derivation method is used by a key-establishment scheme.

that can be supported by HMAC and CMAC, and see SP 800-107[25] for further discussions about the security strength of HMAC).

6.2.3 Symmetric Keys Derived from Passwords

In a number of popular applications, keys are generated from passwords. This is a questionable practice since passwords are usually selected using methods that provide 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 using methods that provide 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 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*).

6.3 Symmetric Keys Produced by Combining (Multiple) Keys and Other Data

When symmetric keys $K_1, \ldots, K_n$ are generated and/or established independently, they may be combined within a key-generating module to form a key K. Other items of data ($D_1, \ldots, D_m$) 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 D_i values need not be kept secret.

The component symmetric keys **shall** be generated and/or established independently (and subsequently protected as necessary) using **approved** methods[27] that support a security strength that is equal to or greater than the targeted security strength of the algorithm or application that will rely on the output key K. Each component key **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 component key **shall not** be used to generate more than one key).

The independent generation/establishment of the component keys $K_1, \ldots, K_n$ 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.

When their use is permitted, $D_1, \ldots, D_m$ **shall** be generated or obtained using methods that ensure their independence from the values of the component keys $K_1, \ldots, K_n$.

The required independence of the component keys from these 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 D_j values, the computation of the D_j values does not depend on any of

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

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

[27] See Sections 4, 6.1, and 6.2.

the K_i values, and knowledge of the D_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 D_j values are secret and the rest of the D_j values (if any) are public, "independence" also means that knowledge of the K_i values and public D_j values yields no information that can feasibly be used to gain insight into the secret D_j values.

Let $K_1, \ldots, K_n$ be the n component keys to be combined to form K. For each K_i (where $i = 1$ to n), let ss_M_i be the maximum security strength that can be supported by the combination of method(s) used to generate K_i and the method(s) used to protect it after generation (e.g., during key transport and/or storage). In particular, assume that an adversary capable of exerting an effort on the order of 2^{ss_Mi} "basic operations" of some sort will be able to compromise those methods and obtain the value of K_i.

The **approved** methods for combining the component keys and other data are:

1. Concatenating two or more keys, i.e.,

$$K = K_1 \mathbin\Vert \ldots \mathbin\Vert K_n.$$

   Notes:

   a. This method requires $n \geq 2$.

   b. The sum of the bit lengths of the n component keys **shall** be equal to *kLen*, the required bit length for K.

   c. The methods used to generate or establish the component keys **shall** be such that the sum of the min-entropies provided by those methods is equal to or greater than the min-entropy required for the resulting key K.

2. Exclusive-Oring one or more symmetric keys and possibly one or more other items of data, i.e.,

$$K = K_1 \oplus \ldots \oplus K_n \oplus D_1 \oplus \ldots \oplus D_m.$$

   Notes:

   a. The length of each component key (K_i) and the length of each D_i **shall** be equal to *kLen*, the required bit length of K.

      This method requires $m \geq 0$, $n \geq 1$ and $n + m \geq 2$.

      - If $m = 0$, then $D_1 \oplus \ldots \oplus D_m$ is an all-zero bit string of bit length *kLen.*
      - If $m = 1$, then $D_1 \oplus \ldots \oplus D_m$ is just D_1.
      - If $n = 1$, then $K_1 \oplus \ldots \oplus K_n$ is just K_1 and $D_1 \oplus \ldots \oplus D_m$ **shall** be a non-zero bit string (in particular, m **shall** be at least 1 in this case).

   b. The methods used to generate or establish the component keys **shall** be such that at least one of those methods provides min-entropy equal to or greater than the min-entropy required for the resulting key K.

3. A key-extraction process, i.e.,

$$K = \mathrm{T}(\text{HMAC-}hash(salt, K_1 \,||\, \ldots || \, K_n \,||\, D_1 \,||\, \ldots || \, D_m), kLen).$$

Notes:

a. HMAC-*hash* **shall** be an implementation of HMAC (as specified in FIPS 198, using an **approved** hash function *hash*) with a security strength that meets or exceeds the targeted security strength of the algorithm or application that will rely on the resulting key K (see SP 800-57, part 1).

b. The *salt* is a secret or non-secret value used as the HMAC key and must be known by all entities using this key-extraction process to obtain the same value of K.

c. This method requires $n \geq 1$. If $n = 1$, then $K_1 \,||\, \ldots || \, K_n$ is just K_1.

d. This method requires $m \geq 0$. If $m = 0$, then $D_1 \,||\, \ldots || \, D_m$ is a null string; if $m = 1$, then $D_1 \,||\, \ldots || \, D_m$ is just D_1.

e. T is the truncation function defined in Section 2.3.

f. The length of the output block of the hash function used with HMAC **shall** be at least *kLen* bits, the required bit length for K.

g. The methods used to generate or establish the component keys **shall** be such that the sum of the min-entropies provided by those methods is equal to or greater than the min-entropy required for the output K; the sum of the min-entropies provided by those methods **should** be equal to or greater than twice the min-entropy required for the resulting key K.

h. Alternative orderings are permitted when forming the concatenation of keys and data (including interleaving the keys and data), but the ordering must be known by all entities computing the value of K.

i. The security strength of the key formed from combining multiple keys and data is subject to the considerations discussed in Section 3.3.

6.4 Distributing Symmetric Keys

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 symmetric 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 by the application or algorithm relying on the symmetric key). The requirements for the output of a key from a cryptographic module are discussed in FIPS 140.

6.5 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 through 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 appropriate method in Section 6 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 6.2.3), 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 6.2.3), that password **shall** be changed prior to the generation of the new key.

References

FIPS 140	National Institute of Standards and Technology (2019) *Security Requirements for Cryptographic Modules*. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-3. https://doi.org/10.6028/NIST.FIPS.140-3
	National Institute of Standards and Technology (2001) *Security Requirements for Cryptographic Modules*. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 140-2, Change Notice 2 December 03, 2002. https://doi.org/10.6028/NIST.FIPS.140-2
FIPS 140 IG	National Institute of Standards and Technology, Canadian Centre for Cyber Security, *Implementation Guidance for FIPS 140-2 and the Cryptographic Module Validation Program*, [Amended]. 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, DC), Federal Information Processing Standards Publication (FIPS) 180-4. https://doi.org/10.6028/NIST.FIPS.180-4
FIPS 186	National Institute of Standards and Technology (2019) Digital Signature Standard (DSS). (U.S. Department of Commerce, Washington, DC), Draft Federal Information Processing Standards Publication (FIPS) 186-5. https://doi.org/10.6028/NIST.FIPS.186-5-draft
FIPS 197	National Institute of Standards and Technology (2001) *Advanced Encryption Standard (AES)*. (U.S. Department of Commerce, Washington, DC), 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, DC), Federal Information Processing Standards Publication (FIPS) 198-1. https://doi.org/10.6028/NIST.FIPS.198-1
FIPS 202	National Institute of Standards and Technology (2015) *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. (U.S. Department of Commerce, Washington, DC), Federal Information Processing Standards Publication (FIPS) 202. https://doi.org/10.6028/NIST.FIPS.202
SP 800-38A	Dworkin MJ (2001) *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. (National Institute of Standards and Technology, Gaithersburg, MD), 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, MD), 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, MD), 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, MD), 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, MD), 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, MD), NIST Special Publication (SP) 800-56C, Rev. 1. https://doi.org/10.6028/NIST.SP.800-56Cr1

SP 800-57-1 Barker EB (2019) *Recommendation for Key Management: Part 1 – General*. (National Institute of Standards and Technology, Gaithersburg, MD), Draft NIST Special Publication (SP) 800-57 Part 1, Rev. 5. https://doi.org/10.6028/NIST.SP.800-57pt1r5-draft

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, MD), 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, MD), Draft NIST Special Publication (SP) 800-71. 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, MD), 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, MD), 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, MD), Draft NIST Special Publication (SP) 800-90C. 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, MD), 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, MD), 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, MD), 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, MD), 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, MD), 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, MD), NIST Special Publication (SP) 800-185. https://doi.org/10.6028/NIST.SP.800-185.

Appendix A: 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.
13. 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.

Revision 2 was made in 2020:

1. Section 2.1: Added definitions for key-derivation function, key-derivation method, key-derivation procedure, key expansion, key extraction, key-wrapping key, MAC key,

message authentication code, nonce, randomness extraction, Recommendation, salt and security function.

Modified: Cryptographic key and support a security strength.

2. Section 2.3: Added *B*, *bLen*, *K*, *kLen*, the max function, and ss_M_i. Modified the truncation function.
3. Section 3.2: : "FIPS 140-**approved**" was changed to "FIPS 140-validate" (twice).
4. Section 3.3, last line: "reduced because of" has been changed to "determined by."
5. Section 4: Modified formula (1) to include additional uses for the resulting bit string, now referring to it as *B* rather *K*. Modified many of the paragraphs for additional clarity.
6. Section 5.1: Revised paragraphs 2 and 3 for further clarity and accuracy.
7. Section 6: Added a 4th bullet about deriving a key during key agreement using a two-step procedure.
8. Section 6.1: Revised the first paragraph.
9. Section 6.2: Combined Sections 6.3, 6.4, and 6.5 of the previous version to address key derivation. Old Section 6.3 is now Section 6.2.1; old Section 6.4 is now Section 6.2.2; and old Section 6.5 is now Section 6.2.3.

   Section 6.2 is now an introductory section for key derivation that includes a list of inputs that could be used for key derivation and a discussion of key-derivation methods.
10. New Section 6.3 (old Section 6.6) on combining multiple keys:
    a. The notation for the other data items has been changed from V_i to D_i to avoid a conflict with the use of V_i in Section 3.4.
    b. Additional discussion on combining key components has been added.
    c. Methods 2 and 3 from the previous version have been combined into the new Method 2.
    d. A new method using HMAC has been added as (a new) Method 3.
    e. Guidance for using each method has been further clarified.
11. New Section 6.4: Moved from the old Section 6.2.
12. References are now an independent section instead of an Appendix.
13. References: FIPS 202 has been added.
14. Appendix B was renamed as Appendix A.