Draft NIST Special Publication 800-71

Recommendation for Key Establishment Using Symmetric Block Ciphers

Elaine Barker
William C. Barker

COMPUTER SECURITY

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

Draft NIST Special Publication 800-71

Recommendation for Key Establishment Using Symmetric Block Ciphers

Elaine Barker
Computer Security Division
Information Technology Laboratory

William C. Barker
Dakota Consulting, Inc.

June 2018

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 the National Institute of Standards and Technology (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 800-71
Natl. Inst. Stand. Technol. Spec. Publ. 800-71, 90 pages (June 2018)
CODEN: NSPUE2

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.

Public comment period: *July 2, 2018* through *September 28, 2018*

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_800-71@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

This recommendation addresses the protection of symmetric keying material during a key establishment that uses symmetric-key cryptography for key distribution. The objective is to provide recommendations for reducing exposure to the unauthorized disclosure of the keying material and detecting its unauthorized modification, substitution, insertion or deletion. The Recommendation also addresses recovery in the event of detectable errors during the key-distribution process. Wrapping mechanisms are specified for encrypting keys, binding key control information to the keys and protecting the integrity of this information.

Keywords

algorithm; authentication; block cipher; key distribution; key establishment; key generation; key management; key translation; key wrapping; message authentication code; symmetric key

Acknowledgements

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by their colleagues at NIST and the members of the ASC X9 working group that developed the standards upon which this Recommendation is based: American National Standard (ANS) X9.17, *Financial Institution Key Management (Wholesale)*, and ANS X9.28, *Financial Institution Multiple Center Key Management (Wholesale)*.

NOTE FOR REVIEWERS

This document, SP 800-71, addresses the use of symmetric block ciphers as key-establishment mechanisms.

The authors acknowledge that most current key-management systems are based on asymmetric cryptography (e.g., a Public Key Infrastructure). However, concerns associated with the projected consequences of emerging quantum computing technology for the security of existing asymmetric algorithms (see NISTIR 8105[1]) suggest a potential for some organizations to reconsider and, on a case-by-case basis, reverting to key establishment based on symmetric cryptography. Given the currently limited nature of guidance on the topic, it seems prudent to describe symmetric key-establishment techniques and security considerations.

Symmetric-key-based key establishment may also be implemented beneath an asymmetric-key-based structure to establish symmetric keys in a hierarchy after the top-most key in the hierarchy has been established using asymmetric key-establishment techniques.

Reviewers are encouraged to provide comments on any aspect of this special publication. Of particular interest are comments on the understandability and usability of the guideline. Your feedback during the public comment period is essential to the document development process and is greatly appreciated.

[1] *Report on Post-Quantum Cryptography*.

Executive Summary

Symmetric-key cryptography requires all originators and consumers of specific information secured by symmetric functions to share a secret key. This is in contrast to asymmetric-key, or public key, cryptography that requires only one party participating in a transaction to know a private key and permits the other party or parties to know the corresponding public key. Symmetric-key cryptography is generally much more computationally efficient than public key cryptography, so it is most commonly used to protect larger volumes of information such as the confidentiality of data in transit and in storage. Asymmetric cryptography is more commonly used for the establishment of an initial symmetric key using key-agreement or key-transport techniques. There are circumstances however, such as the discovery or emergence of serious vulnerabilities of common public key algorithms to technological attacks, that may motivate individuals and organizations to use symmetric-key cryptography for source authentication, data integrity and key-establishment purposes.

This Recommendation addresses the protection of symmetric keying material during key establishment using symmetric-key algorithms. The objective is to reduce the potential for unauthorized disclosure of the keying material and enable the detection of unauthorized modification, substitution, insertion and deletion of that keying material. The Recommendation also addresses recovery in the event of detectable errors during the key-establishment process.

Several key-establishment architectures are described. These include:

- Key establishment among communicating groups that share a key-wrapping key,
- The distribution of keys by key generation and distribution centers to their subscribers,
- The use of translation centers for the protected distribution of keys generated by one subscriber for distribution to one or more other subscribers, and
- Multiple-center-based environments for key establishment between or among organizational domains.

The Recommendation does not specify protocols for key establishment (e.g., Kerberos, S/MIME, and DSKPP). It does, however, suggest key-establishment communication options and transaction content that **should** be accommodated by key-establishment protocols.

This Recommendation covers both the manual and automated management of symmetric keying material for the federal government using symmetric-key techniques. The Recommendation **should** be used in conjunction with the SP 800-57[2] series of documents and SP 800-152[3] for the management of keying material, including:

- Control during the life of the keying material to prevent unauthorized disclosure, modification or substitution;

[2] SP 800-57: *Recommendation for Key Management, Part 1: General, Part 2: Best Practices for Key Management, and Part 3: Application-Specific Key Management Guidance.*

[3] SP 800-152: *A Profile for U.S. Federal Cryptographic Key Management Systems (CKMS).*

- Establishing communicating groups;
- Secure distribution of keying material to permit interoperability among communicating groups;
- Ensuring the integrity of keying material during all phases of its life, including its establishment (which includes generation and distribution), storage, entry, use, and destruction;
- Recovery in the event of a failure of the key-establishment process or when the integrity of the keying material is in question; and
- Auditing the key-management processes.

Important considerations that apply to the selection of a key-management approach include:

- The exposure of a key by any entity having access to that key compromises all data protected by that key;
- The more entities that share a key, the greater the probability of exposure of that key to unauthorized entities;
- The longer that a key is used, the greater the chance that it will become known by unauthorized parties during its use;
- The greater the amount of data that is protected by the key, the greater the amount of data that is exposed if the key is compromised;
- It is essential that the source of a secret or private key is trustworthy, and that a secure channel be used for key distribution; and
- The key used to initiate a keying relationship must be obtained through a secure channel, often using an out-of-band process.

This Recommendation provides general guidance for the establishment of symmetric keys. It is intended to be a general framework within which system-specific protocols may be applied. Public key cryptography is mentioned only as an alternative method for establishing an initial keying relationship for a communicating group.

TABLE OF CONTENTS

1. Introduction

Symmetric-key cryptography employs cryptographic algorithms that require both the sending and receiving parties to protect communications using the same secret key. This is distinct from asymmetric-key (i.e., public key) cryptography in which the parties have pairs of keys – a private key known only to the key pair owner, and a public key that may be known by anyone. Section 3 of SP 800-175B[4] discusses the use of these two algorithm types, including the pros and cons of each, namely that:

- Symmetric-key cryptography is generally much less computationally intensive than asymmetric-key cryptography.
- Digital signatures generated using asymmetric-key algorithms provide better source authentication properties than can be provided by symmetric-key algorithms.
- The number of keys required to initiate and maintain cryptographic keying relationships is much higher for symmetric-key cryptography than for asymmetric-key cryptography.

As a result of these characteristics, recent key-management schemes have used symmetric-key cryptography for the encryption and integrity protection of data-at-rest and data-in-transit (i.e., stored or communicated data), and asymmetric-key cryptography to establish the symmetric keys for data-in-transit and for source authentication and integrity protection using digital signatures.[5]

Recent concerns associated with the projected consequences of emerging quantum-computing technology for the security of existing asymmetric algorithms (see NISTIR 8105[6]) suggest a potential federal government requirement for the reconsideration of, and possible reversion to, the use of symmetric-key cryptography. Keys protected using currently **approved** asymmetric-key algorithms[7] can, therefore, be expected to become known by adversaries once quantum computers become available. In contrast, the impact on symmetric-key algorithms will not be as drastic; doubling the size of the key will be sufficient to preserve security. Symmetric-key algorithms and hash functions with sufficiently large output should be usable in a quantum era.

Research is in progress to develop quantum-resistant asymmetric-key algorithms.[8] However, replacing the currently used asymmetric-key algorithms with quantum-resistant asymmetric-key algorithms can be expected to not really begin until about 2020 and not be completed until the 2030s.

Where the security of information is very important, and the security of information currently being protected by asymmetric-key algorithms needs to be maintained for more than a few years,

[4] NIST SP 800-175B, *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms*, August 2016.

[5] Note that symmetric key management is used in some applications such as over-the-air rekeying of digital radios. See Section 7 of SP 800-57 Part 3, *Recommendation for Key Management Part 3: Application-Specific Key Management Guidance* and Kerberos.

[6] NISTIR 8105, *Report on Post-Quantum Cryptography*, April 2016.

[7] Algorithms based on the use of difficult problems such as integer factorization, discrete logarithms, and elliptic-curve discrete-logarithms.

[8] See https://csrc.nist.gov/Projects/Post-Quantum-Cryptography.

moving away from the protection of symmetric keys by asymmetric-key algorithms **should** be initiated as soon as practical. The protection of symmetric keys using symmetric key-wrapping schemes and replacing asymmetric digital signature schemes with symmetric-key message authentication schemes is one approach to replacing public key cryptographic key management in the relatively near term.

The subject of this Recommendation is the set of security considerations associated with the use of symmetric-key algorithms for key establishment. It addresses the protection of symmetric keying material during key establishment to prevent unauthorized disclosure of the keying material and to detect unauthorized modification, insertion and deletion. This Recommendation also addresses the recovery of keys in the event of detectable errors during the key-establishment process. Several high-level key-establishment strategies are presented.

While specific protocols (e.g., Kerberos[9], S/MIME,[10] and DSKPP[11]) are not specified in this Recommendation, this document does suggest key-establishment transaction content and options that **should** be accommodated by key-establishment protocols. A minimum set of requirements for constructing an audit trail of the key establishment process is provided in SP 800-152.

Note that conformance to this Recommendation does not guarantee security. Because the Recommendation is protocol-independent, the specific protocol employed for key-establishment purposes needs to be analyzed for adequacy within the context of an organization's security goals. Several key-establishment approaches are described in this document. Although the strategies described include several key-establishment environments, the Recommendation does not preclude the use of other symmetric-key management approaches.

1.1 Scope

Although this Recommendation describes the automated disribution of symmetric keying material using symmetric-key techniques in automated environments, manual distribution is discussed as well.

This Recommendation focuses primarily on strategies for the management of keys prior to their use for protecting data communications. However, the Recommendation, in conjunction with the SP 800-57 series of documents and SP 800-152 contain the minimum requirements for the management of keying material throughout its lifecycle, including:

- Control during the life of the keying material to prevent unauthorized disclosure, modification or substitution;
- Establishing communicating groups;
- The secure distribution of keying material to permit interoperability among communicating groups;

[9] See Section 6 of SP 800-57 Part 3, *Recommendation for Key Management Part 3: Application-Specific Key Management Guidance*.
[10] S/MIME: Secure Multipurpose Internet Mail Extensions.
[11] DSKPP: Dynamic Symmetric Key Provisioning Protocol.

- Ensuring the integrity of keying material during all phases of its life, including its establishment (which includes generation and distribution), storage, entry, use, and destruction;
- Recovery in the event of a failure of the key-establishment process or when the integrity of the keying material is in question; and
- Auditing the key-management processes.

The scope of this document encompasses the use of only symmetric-key block-cipher algorithms (e.g., FIPS 197[12]) and algorithms used to generate Message Authentication Codes (MACs) using either block-cipher algorithms or using hash functions (e.g., FIPS 180-4[13] and FIPS 202[14]). The use of asymmetric-key (i.e., public-key) techniques for key establishment is mentioned only as an alternative method for establishing an initial keying relationship.

1.2 Content and Organization

The remainder of this Recommendation is organized as follows:

Section 2 provides definitions and common abbreviations.

Section 3 provides general symmetric key-management fundamentals, including uses for symmetric keys, some application considerations, symmetric algorithms and key types, key-distribution using symmetric-key techniques, and a discussion of key hierarchies for storage and communications applications.

Section 4 describes general architectural considerations for the establishment of symmetric keys – both center-based key establishment and key establishment among communicating groups.

Section 5 discusses key-establishment communications, including general communication requirements, key names and key labels, message content and handling, authentication codes in key-establishment messages and key revocation and destruction.

Appendix A contains example scenarios, and Appendix B lists document references.

[12] FIPS 197, *Advanced Encryption Standard (AES)*, November 26, 2001.

[13] FIPS 180-4, *Secure Hash Standard (SHS)*, March 2012.

[14] FIPS 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, August 4, 2015.

2. Definitions and Common Abbreviations

2.1 Definitions

Acknowledgement information	Information sent to acknowledge the receipt of a communication without errors.
Advanced Encryption Standard	The encryption algorithm specified by FIPS 197, *Advanced Encryption Standard*.
Agent	See multiple-center agent.
Approved	FIPS-**approved** or NIST-recommended. An algorithm or technique that is either 1) specified in a FIPS or NIST Recommendation, or 2) specified elsewhere and adopted by reference in a FIPS or NIST Recommendation.
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.
Asymmetric-key cryptography	Cryptography that uses pairs of keys: public keys that may be widely disseminated and private keys that are authorized for use only by the owner of the key pair and known only by the owner and possibly a trusted party that generated them for the owner.
Authenticated data	Data that is accompanied by a valid message authentication code that is used to verify its source and that the data is identical to that for which the message authentication code was computed.
Authenticated encryption keys (AEKs)	Keys used to provide both confidentiality and integrity protection for the target data using the same key. Block cipher modes for using AEKs are specified in SP 800-38C[15] and SP 800-38D.[16]
Authentication	A process that provides assurance of the source and integrity of information that is communicated or stored.
Authentication algorithm	A cryptographic function that is parameterized by a symmetric key. The algorithm acts on input data (called a "message") of variable length to produce an output value of a specified length.

[15] SP 800-38C, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*.

[16] SP 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*.

	The output value is called the message authentication code (MAC) of the input message.
Authentication key	A symmetric key used to generate a message authentication code on a message. See Data Authentication Key (DAK).
Authenticity	The property of being genuine, verifiable and trusted; confidence in the validity of a transmission, a message, or message originator.
Automated	Using an electronic method rather than a manual method. In most cases, no human intervention is required.
Automated key establishment	The process by which cryptographic keys are securely distributed among cryptographic modules using automated methods (e.g., key transport and/or key agreement protocols).
Bi-directional (communications)	As used in this Recommendation, the same symmetric key can be used for both protecting (e.g., encrypting) sensitive data to be sent to one or more other entities and for processing (e.g., decrypting) protected data received from other entities sharing the key. Contrast with uni-directional (communications).
Block cipher	A symmetric-key cryptographic algorithm that transforms one block of information at a time using a cryptographic key. For a block cipher algorithm, the length of the input block is the same as the length of the output block.
Checksum	A value that (a) is computed by a function that is dependent on the contents of a data object and (b) is stored or transmitted together with the object, for detecting changes in the data.
Ciphertext	Data in its encrypted form.
Cloud computing facility	A facility that provides ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.
Compromise	The unauthorized disclosure, modification or use of sensitive data (e.g., keying material and other security-related information).
Confidentiality	The property that sensitive information is not disclosed to unauthorized individuals, entities, or processes.

Communicating group	Two or more logical entities that exchange data using a set of common keying material. Each communicating group has different keying material. An entity and a center participating in a key-establishment transaction do not constitute a communicating group.
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 key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. Examples include: 1) The transformation from plaintext to ciphertext and vice versa for a given cryptographic algorithm, or 2) The Message Authentication Code for given data and cryptographic algorithm.
Cryptoperiod	The time span during which a specific key is authorized for use or in which the keys for a given system may remain in effect.
Data Authentication Key (DAK)	A key used for the computation of MACs in order to provide assurance of content integrity and (some level of) source authentication for cryptographically protected information.
Data Encrypting Key (DEK)	A key used for the encryption of data.
Data Key (DK)	A key used to encrypt and decrypt data, or to authenticate data.
Decryption	The process of transforming ciphertext into plaintext using a cryptographic algorithm and key.
Encryption	A process of transforming plaintext into ciphertext using a cryptographic algorithm and key.
Entity	An individual (person), organization, device, or process.
Error report information	The information in a message that reports the error that was found in a previously received message.
Hash function	A function that maps a bit string of arbitrary length to a fixed-length bit string. **Approved** hash functions satisfy the following properties: 1. (One-way) It is computationally infeasible to find any input that maps to any pre-specified output, and

	2. (Collision resistant) It is computationally infeasible to find any two distinct inputs that map to the same output.
Impact level	The magnitude of harm that can be expected to result from the consequences of unauthorized disclosure of information, unauthorized modification of information, unauthorized destruction of information, or loss of information or information system availability.
Internet Engineering Task Force (IETF)	A large, open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.
Initialization vector (IV)	A vector used in defining the starting point of a cryptographic process.
Key	See Cryptographic key.
Key agreement	A key-establishment procedure where the resultant keying material is a function of information contributed by two or more participants, so that an entity cannot predetermine the resulting value of the keying material independently of any other entity's contribution.
Key Derivation Key (KDK)	Keys used to derive DEKs, DAKs, AEKs. and other KDKs Symmetric-key methods for key derivation are specified in SP 800-108.[17] KDKs are not used to derive KWKs.
Key Distribution Center (KDC)	Used to generate and distribute keys to entities that need to communicate with each other but may not share keys except with the center.
Key establishment	The process by which a key is securely shared between two or more entities, either by transporting a key from one entity to another (key transport) or deriving a key from information contributed by the entities (key agreement).
Key-establishment transaction	An instance of establishing secret keying material among entities. A transaction will require multiple protocol messages between two or more entities.

[17] SP 800-108, *Recommendation for Key Derivation Using Pseudorandom Functions.*

Key-generation request information	Information necessary to request the generation of cryptographic keys.
Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., IVs) during the entire life cycle of the keys, including their generation, storage, establishment, entry and output, and destruction.
Keying material	The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships.
Keying relationship	The state existing between entities when they share at least one symmetric key.
Key-transfer information	Information used to distribute one or more keys to a recipient.
Key Translation Center (KTC)	Used to unwrap keying material sent by one subscriber using a key-wrapping key shared with that subscriber, and to rewrap the same keying material using a different key-wrapping key shared with a different subscriber.
Key transport	A manual or automated key-establishment procedure whereby one entity (the sender) selects and distributes the key to another entity (the receiver).
Key type	As used in this Recommendation, a key categorized by its properties and uses: key-wrapping key, data authentication key, data encryption key or key-derivation key.
Key unwrapping	A method of removing the cryptographic protection on keys that was applied using a symmetric-key algorithm and key-wrapping key.
Key wrapping	A method of cryptographically protecting keys that provides both confidentiality and integrity protection for the wrapped keying material using a symmetric-key algorithm and a key-wrapping key.
Key Wrapping Key (KWK)	A key used exclusively to wrap and unwrap (e.g., encrypt, decrypt and integrity protect) other keys.
Layer 1 key	The top-most layer in a (possible) hierarchy of keys of a keying relationship.

Manual distribution	A non-automated means of transporting cryptographic keys by physically moving a device or document containing the keying material.
Master/recipient relationship	As used in this Recommendation, one (or more) members of a communicating group (i.e., masters) are allowed to generate keying material and distribute it to all other members of the group, while other members (i.e., recipients) are only allowed to receive keying material. Contrast with a peer relationship.
Message	The information transferred from one entity to another using communication protocols. This Recommendation identifies information to be included in a message but does not specify the format of that message.
Message Authentication Code (MAC)	A cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data.
Mode (of operation)	A set of rules for operating on data with a cryptographic algorithm and a key; often includes feeding all or part of the output of the algorithm back into the input of the next iteration of the algorithm, either with or without additional data being processed.
Multicast transmission	A transmission that communicates a set of information from one sender to multiple recipients simultaneously.
Multiparty control	A process that uses two or more separate entities (usually persons) operating in concert to protect sensitive functions or information. No single entity is able to access or use the materials, e.g., cryptographic keys.
Multiple-center agent	A center within a multiple-center group through which a subscriber obtains multiple-center key-establishment services.
Multiple-center group	A set of two or more centers that have agreed to work together to provide cryptographic keying services to their subscribers.
Party	Any entity, center or multiple-center agent.
Peer relationship	As used in this Recommendation, all members of a communicating group are allowed to generate or otherwise obtain keying material for distribution to the other members of the group. Contrast with a master/recipient relationship.

Protocol	A special set of rules used by two or more communicating entities that describe the message order and data structures for information exchanged between the entities.
Public key cryptography	See asymmetric-key cryptography.
Plaintext	Unencrypted (unenciphered) data.
Recipient	The entity that receives a communication.
Revocation	As used in this Recommendation, the process of permanently terminating the valid use of a key to apply cryptographic protection (e.g., wrap keying material, encrypt data or generate a MAC).
Revocation-confirmation information	Information provided to confirm that keying material has been destroyed as requested.
Revocation-request information	Information indicating the keys to be revoked and destroyed.
Secure channel	As used in this Recommendation, 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 operations) that is required to break a cryptographic 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**.
Should	This term is used to indicate an important recommendation. Ignoring the recommendation could result in undesirable results. Note that **should** may be coupled with **not** to become **should not**.
Source authentication	A process that provides assurance of the source of information.
Split knowledge	A process by which a cryptographic key is split into *n* key components, each of which provides no knowledge of the original

	key. The components can be subsequently combined to recreate the original cryptographic key.
Subscriber	An entity that has a keying relationship with a center or agent of a multiple-center group.
Symmetric key	A single cryptographic key that is used with a symmetric-key algorithm.
Symmetric-key algorithm	A cryptographic algorithm that uses a single secret key for a cryptographic operation and its complement (e.g., encryption and decryption).
Symmetric-key cryptography	Cryptography that uses the same key for both applying cryptographic protection (e.g., encryption or computing a MAC) and removing or verifying that protection (e.g., decryption or verifying a MAC).
Target data	As used in this Recommendation, data, other than keys, that are afforded cryptographic protection.
Time-variant parameter	A time-varying value that has (at most) an acceptably small chance of repeating (where the meaning of "acceptably small" may be application specific).
Transaction	See Key-establishment transaction.
Transaction-authentication key	A key generated specifically for the key-establishment transaction that is used to generate message authentication codes for the protocol messages in that transaction.
Translation	The process performed by a center to unwrap keying material received from a sending entity (a subscriber or a center in a multiple-center group) using a key-wrapping key shared with that entity and then rewrapping the same keying material using a different key-wrapping key shared with the next recipient of the wrapped keying material (a different subscriber or a different center in the multiple-center group).
Translation-request information	Information provided to a center to request the translation of keying material contained in the request for a subscriber.
Uni-directional (communications)	As used in this Recommendation, a different symmetric key is always required for cryptographically protecting (e.g., encrypting) sensitive data to be sent to another entity than is required when processing (e.g., decrypting) cryptographically

	protected data that is received from that other entity. Contrast with bi-directional (communications).
Wrapping	See Key wrapping

2.2 Common Abbreviations

This section contains abbreviations used in this Recommendation.

AEK	Authenticated Encryption Key.
AES	Advanced Encryption Standard.
DAK	Data Authentication Key.
DEK	Data Encrypting Key.
DK	Data Key.
FIPS	Federal Information Processing Standard.
KDC	Key Distribution Center.
KDK	Key Derivation Key.
KWK	Key Wrapping Key.
KTC	Key Translation Center.
MAC	Message Authentication Code.
NIST	National Institute of Standards and Technology.
NISTIR	NIST Internal or Interagency Report.
SP	Special Publication.

3. Symmetric-Key-Management Fundamentals

Symmetric-key algorithms (sometimes called secret-key algorithms) use a single key to both apply cryptographic protection and to remove or check the protection. For example, the key used to encrypt data (i.e., apply protection) is also used to decrypt the encrypted data (i.e., remove the protection); in the case of encryption, the original data is called the plaintext, while the encrypted form of the data is called the ciphertext. The key must be kept secret if the data is to remain protected.

The goals of symmetric-key management are 1) to provide keys and related cryptographic variables (e.g., initialization vectors (IVs)) where they are needed and 2) to keep keys secret. The security of the data protected by these keys is strictly dependent upon the prevention of unauthorized disclosure, modification, substitution, insertion, and deletion of the keys and, as appropriate, other cryptographic variables (e.g., IVs). If these are compromised, the confidentiality and integrity of the protected data can no longer be assured. General key-management guidelines are provided in SP 800-57 Part 1. Basic requirements for Key Management Systems operated by or for the Federal Government are provided in SP 800-152.

3.1 Uses of Symmetric Keys

Symmetric keys are used by block cipher algorithms (e.g., AES) that are used for encryption, key wrapping and/or the generation of message authentication codes. Symmetric keys are also used by hash function-based authentication algorithms (e.g., HMAC[18] and KMAC[19]) for the generation of message authentication codes, and for key derivation and random bit generation.

Encryption is used to provide confidentiality for data. The unprotected form of the data is called plaintext. Encryption transforms the data into ciphertext, and ciphertext can be transformed back into plaintext using decryption. Data encryption and decryption are generally provided using symmetric-key block cipher algorithms. See Section 4.1 of SP 800-175B[20] for more information regarding data encryption.

Key wrapping is a method used to provide confidentiality and integrity protection for keys (and possibly other information associated with the keys) using a symmetric key-wrapping key that is known by both the sender and receiver, and a block cipher algorithm. The wrapped keying material can then be stored or transmitted (i.e., transported) securely. Unwrapping the keying material requires the use of the same algorithm and key-wrapping key that was used during the original wrapping process. See Section 5.3.5 of SP 800-175B for more information on key wrapping.

[18] HMAC is specified in FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC)*.
[19] KMAC is specified in SP 800-185, *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash*.
[20] SP 800-175B, *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms*.

Message authentication codes are used to protect message and data integrity. Message authentication codes are cryptographic checksums on data that use symmetric-key cryptography to detect both accidental and intentional modifications of data. They also provide some measure of source authentication between entities sharing the same key because only entities sharing a key can produce the same message authentication code. See Section 4.2 of SP 800-175B for further information on message authentication codes.

Key derivation is concerned with the generation of a key from secret information, although non-secret information may also be used in the generation process in addition to the secret information. Typically, the secret information is shared among the entities that need to derive the same key for subsequent interactions. The secret information could be a key that is already shared between the entities (i.e., a pre-shared key), or could be a shared secret that is derived during a key-agreement scheme. See Section 5.3.2 of SP 800-175B for more information regarding key derivation.

Cryptography and security applications make extensive use of random numbers and random bits. For cryptography, random values are needed to generate cryptographic keys. There are two classes of random bit generators (RBGs): Non-Deterministic Random Bit Generators (NRBGs), sometimes called true random number (or bit) generators, and Deterministic Random Bit Generators (DRBGs), sometimes called pseudorandom bit (or number) generators. SP 800-90A[21] specifies **approved** DRBG algorithms, based on the use of hash functions and block-cipher algorithms. See Section 4.4 of SP 800-175B for more information regarding random bit generation.

3.2 Application Considerations

Federal agencies are required to comply with FIPS 199[22] and FIPS 200[23] in determining the sensitivity of their applications and data (i.e., the target data) and the impact level associated with any compromise of that data (i.e., Low, Moderate or High impact). When the impact level has been determined, the security strength of the cryptographic algorithms and keys for protecting that data can be determined. **PR:2.3**, **PR:2.4** and **PR:2.5** in SP 800-152 specify the minimum security strengths required for the Low, Moderate and High impact levels, respectively.

Important considerations that apply to the selection of a key-management approach include:

- The exposure of a key by any entity having access to that key compromises all data protected by that key;
- The more entities that share a key, the greater the probability of exposure of that key to unauthorized entities;

[21] SP 800-90A, *Random Number Generation Using Deterministic Random Bit Generator Mechanisms.*
[22] FIPS 199, *Standards for Security Categorization of Federal Information and Information Systems.*
[23] FIPS 200, *Minimum Security Requirements for Federal Information and Information Systems.*

- The longer that a key is used, the greater the chance that it will become known by unauthorized entities during its use;
- The greater the amount of data that is protected by the key, the greater the amount of data that is exposed if the key is compromised;

> The exposure of a key by any entity having access to that key compromises all data protected by that key, and the more entities that share a key, the greater the probability of exposure of that key to unauthorized entities.

- It is essential that the source of a secret or private key is trustworthy, and that a secure channel be used for key distribution; and
- The key used to initiate a keying relationship must be obtained through a secure channel, often using an out-of-band process.

Each of these considerations must be addressed in any application of symmetric-key cryptography.

When using asymmetric cryptography, one entity can make one public key available to other entities and use the corresponding private key in secured communications with those other entities. However, when using symmetric-key cryptography, a different key is often required for each correspondent. Some organizations choose to reduce this cryptographic burden by sending the same symmetric key to multiple correspondents, then using that key in multicast transmissions to, or exchanges with, all parties sharing that symmetric key. Drawbacks to this approach include a loss of privacy and integrity protections within what are effectively cryptographic communities-of-interest, and a loss of cryptographic protection by all members of the community-of-interest if the shared key is compromised. There is also significant management and accounting overhead associated with the distribution, installation, revocation and post-revocation access management for what can be complex combinations of both distinct and overlapping cryptographic communities.

Symmetric-key cryptography is attractive in applications that cannot afford the processing overhead associated with asymmetric cryptography. This is becoming a more important factor, given the rapid growth of the Internet of Things (IoT). Symmetric-key cryptography is an increasingly common choice for Wireless Sensor Networks (WSN), for example, due to the limited processing, storage, and electrical power available to sensors. As of 2018, asymmetric-key encryption, even for key-establishment and integrity protection is impractical for many IoT sensor components. An initial response to this situation has resulted in research to develop "lightweight" block ciphers (see NISTIR 8114[24]) to protect sensor data and control. These "lightweight" block ciphers can be defeated by current personal computers in one to a few hours (see KM in WSN).

[24] NIST 8114, *Report on Lightweight Cryptography*.

> The longer that a key is in use, the greater the chance that it will become known by unauthorized parties while still in use, and the greater the amount of data protected by the key, the greater the amount of data that is compromised if the key is compromised.

Some applications of symmetric-key cryptography reduce the initial key-management overhead by establishing "crypto nets" in which many entities share the same secret key. Although there are cases where operational considerations encourage the adoption of this course, the exposure of any secret key tends to become more likely as the number of entities sharing the secret key increases. Cyber threats, personnel security threats, physical security threats and simple carelessness on the part of any entity that has access to an unencrypted secret key endangers the security of all data protected by that key. This consideration argues in favor of restricting the number of entities that share any given key. Exceptions that can mitigate the effects of this principle are found in isolated environments, such as networks in protected facilities in which no processor that has a secret key is remotely accessible.

> It is essential that the source of a secret or private key be trustworthy; the key used to initiate a keying relationship must be obtained using a secure channel.

For these reasons, keys **shall not** be used indefinitely. The period for which a key is to be used, called a cryptoperiod, is established by policy based on a risk assessment. In any event, symmetric-key management involves not just the initial distribution of keys, but also the distribution of replacements for expired or compromised keys. Key replacement is required at a frequency determined by the cryptoperiod, but emergency replacement is also required when a key in use is compromised. The distribution and accounting requirements imposed by cryptoperiods and emergency key replacement add significantly to key-management overheads. Note that even the management of asymmetric-key pairs imposes a sufficient overhead burden that many organizations seek to minimize when using cryptography. However, the key-management burden is greater in the case of symmetric-key cryptography.

The source of any secret key has the ability to defeat any confidentiality or integrity mechanism for which the key is used. Consequently, keys **shall** be accepted only from sources that can be trusted with all information that is to be protected by cryptography using those keys.

When using asymmetric-key cryptography, a secure communications relationship can be established with a new correspondent simply by making a key-establishment public key available to the new correspondent. In the case of symmetric-key cryptography, a secret key must be securely provided to the new correspondent. This requires either a physical transfer between correspondents, a shared relationship with a center (e.g., a key distribution center) or the establishment of an initial symmetric key using asymmetric key-establishment techniques.

Cloud-computing facilities and other large data repositories that store and/or process information for physically remote customers **should** protect that information while in transit and at rest. Due to its superior processing efficiency, symmetric-key cryptography is used for the encryption of the information, although asymmetric-key cryptography has generally been used for key transport and integrity protection and for the generation of digital signatures. Some cloud-computing facilities and networks serve very large numbers of customers. Secure storage, retrieval, and general management of the symmetric keys is essential to the confidentiality of customer information. It also represents significant key-management overhead. Symmetric keys must never be stored or transferred in unprotected form.

In the past, most distributions of symmetric keys involved a transfer of the keys by human couriers or secure government mail systems. However, as the number of entities using a system grows, the work involved in the distribution of the secret keying material could grow to be prohibitive. The Internet Engineering Task Force (IETF's) provides guidelines for key management in RFC 4107[25], which discusses issues associated with manual versus automated key distribution, as well as best practices for key management. Consistent with RFC 4107's conclusion that, in general, automated key management **should** be employed, this Recommendation focuses primarily on automated key-establishment schemes. However, for any cryptographic key-management scheme that is solely dependent on symmetric-key cryptography for key establishment, the initial distribution of keys without the use of asymmetric-key algorithms must be manual. This is a significant cost constraint and introduces architectural complexity as the size of the supported organization increases.

3.3 Symmetric Algorithm and Key Types

NIST has **approved** several basic cryptographic algorithms and "modes" for using them.

- Block cipher algorithms (e.g., AES and TDEA[26]) that are used in specified modes to perform encryption/decryption, message authentication and integrity protection, key wrapping, key derivation and random bit generation.
- Hash functions (algorithms) that can be used to provide message authentication and integrity protection, key derivation and random bit generation. The methods for providing these services can be considered as hash function modes, although that term is not normally used in relation to hash functions.

Several types of keys are used in symmetric-key cryptography.

[25] RFC 4107, *Guidelines for Cryptographic Key Management.*

[26] Although TDEA is currently an approved algorithm, its use is being discouraged because of security considerations (see SP 800-131A and the NIST announcement for using TDEA).

- Key wrapping keys (KWKs) are used to wrap (i.e., encrypt and integrity protect) other keys, including other KWKs. KWKs are used with a block cipher algorithm as specified in SP 800-38F.[27]
- Data encryption keys (DEKs) are used to encrypt data other than keys (i.e., the target data). Block cipher modes for using DEKs are specified in SP 800-38A[28], the addendum to SP 800-38A[29], SP 800-38E[30] and SP 800-38G.[31]
- Data authentication keys (DAKs) are used to generate message authentication codes (MACs) that provide integrity protection and (some measure of) source authentication for the target data. Block cipher modes for generating and verifying MACs are specified in SP 800-38B[32] and SP 800-38D.[33] Hash-based techniques for generating and verifying MACs are specified in FIPS 198[34] and SP 800-185.
- Authenticated encryption keys (AEKs) are used to provide both confidentiality and integrity protection for the target data using the same key. Block cipher modes for using AEKs are specified in SP 800-38C[35] and SP 800-38D.
- Key Derivation Keys (KDKs) can be used to derive DEKs, DAKs, AEKs and other KDKs. Symmetric-key methods for key derivation are specified in SP 800-108.[36] KDKs **shall not** be used to derive KWKs.

DEKs, DAKs and AEKs are collectively called data keys (DKs).

3.4 Key Distribution Using Symmetric-Key Techniques

Keying material (i.e., keys and other cryptographic variables, such as IVs) **shall** either be distributed manually (see Section 3.4.1) or using appropriate automated distribution methods (see Section 3.4.2) before secure transactions begin using those keys. Keys, all other cryptographic

---

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

[28] SP 800-38A, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques.*

[29] SP 800-38A Addendum, *Recommendation for Block Cipher Modes of Operation: Tree Variants of Ciphertext Stealing for CBC Mode.*

[30] SP 800-38E, *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices.*

[31] SP 800-38G, *Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption.*

[32] SP 800-38B, *Recommendation for the Block Cipher Mode of Operation: the CMAC Mode for Authentication.*

[33] SP 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC,* SP 800-38D.

[34] FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC).*

[35] SP 800-38C, *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality.*

[36] SP 800-108, *Recommendation for Key Derivation Using Pseudorandom Functions.*

variables (where needed), and accompanying documentation **shall** be protected throughout the distribution process.

Keys **shall not** be used operationally to apply cryptographic protection (e.g., encrypt) prior to sending and/or receiving acknowledgments of successful receipt or if a compromise is suspected. Procedures to follow up and resolve distribution irregularities **shall** be in place (e.g., included in a Key Management Practices Statement as described in SP 800-57, Part 2.[37].

3.4.1 Manual Distribution

When manual methods are used to distribute cryptographic keying material, that material **shall** be distributed using couriers, registered mail, or an equivalent distribution service in which the delivery agent is trusted by both the sending and receiving entities, with the recipients required to identify themselves to the delivery agent and provide an appropriate receipt upon delivery. The keys **shall** be transported on a medium that, together with the physical distribution method, provides the required confidentiality and integrity protection for the keys.

Electronic media (e.g., smart cards, flash drives, or key loader devices) **should** be used during manual distribution. If keys or other cryptographic variables are printed (instead of being distributed using electronic media), provision **shall** be made to protect the keying material from unauthorized disclosure or replacement (e.g., using uniquely identified, tamper-detecting packaging). Whether using electronic media or printed material during delivery, the delivery receipt **shall** identify the source of the keying material, the delivery agent, the recipient, and indicate the state of the received media (e.g., no tampering detected, valid authentication codes, etc.).

For environments where the FIPS 199 impact level associated with the data to be protected by the keying material to be distributed is High, multiparty control and/or split knowledge **shall** be employed when keys are distributed in plaintext form.

Distribution procedures **shall** ensure that:

(1) The distribution of keys and any other variables is authorized;

(2) The keying material has been received by the authorized recipient; and

(3) The key has not been disclosed, modified or replaced in transit.

The distributor (i.e., the source of the keying material) and receiver of the manually distributed keys **shall** identify (to each other) those individuals who are authorized to originate, receive and change keys and **shall not** reassign or delegate such responsibilities without proper notice.

---

[37] SP 800-57, Part 2: *Recommendation for Key Management: Part 2: Best Practices for Key Management Organizations.*

3.4.2 Automated Distribution

Automated key distribution is the electronic transmission of cryptographic keys (and, where needed, other cryptographic variables such as IVs) via a communication channel (e.g., the Internet). This requires the prior distribution of an initial key-wrapping key (KWK) and an authentication key (i.e., a DAK), either manually (see Section 3.4.1) or using asymmetric key-establishment techniques (e.g., the key agreement or key transport schemes specified in SP 800-56A or SP 800-56B). The KWK and DAK may then be used to distribute all key types discussed in Section 3.3.

Keying material distributed after the initial KWK and DAK have been established **shall** be wrapped with a KWK shared between communicating entities[38] in key-establishment messages defined using a protocol that provides confidentiality, integrity protection assured delivery, and replay protection; the content of the protocol message **shall** be integrity protected using a DAK[39] (see Section 5.4). The recipient(s) **shall** unwrap the protected keys and verify their source and integrity before any cryptographic process can begin for communications using the transported key(s). If a recipient has multiple KWKs that may be used to unwrap the received keys, information **shall** be available to identify the KWK to be used (e.g., sent with the transported keying material) (see Section 5.2). Likewise, if multiple DAKs are available, a method **shall** be available to indicate the DAK used.

An SP 800-38F-compliant key-wrapping algorithm **shall** be used with a KWK for wrapping keys for automated key distribution. The key-wrapping algorithm **shall** use an **approved** symmetric encryption algorithm (i.e., AES) for wrapping one or more keys during the same key-wrapping process. Keys being wrapped may be either KWKs, KDKs, DEKs, DAKs or AEKs. The algorithm and key size used to perform the key wrapping **shall** provide security equal to or greater than the security strength to be provided to any data to be subsequently protected by the wrapped keys.

A means of protection against replay **shall** be provided in a key-establishment protocol. The use of time-variant parameters may be used to afford this protection. A nonce is a time-varying value that has (at most) an acceptably small chance of repeating (where the meaning of "acceptably small" may be application specific). See Section 5.4 of SP 800-56A or SP 800-56B for more information on nonces.

3.5 Key Hierarchies

A hierarchy of keys is often used when symmetric-key cryptography is employed for communications and storage applications.

[38] Either the initial KWK or a KWK subsequently distributed between the communicating entities.

[39] Either the initial DAK or a DAK subsequently distributed between entities.

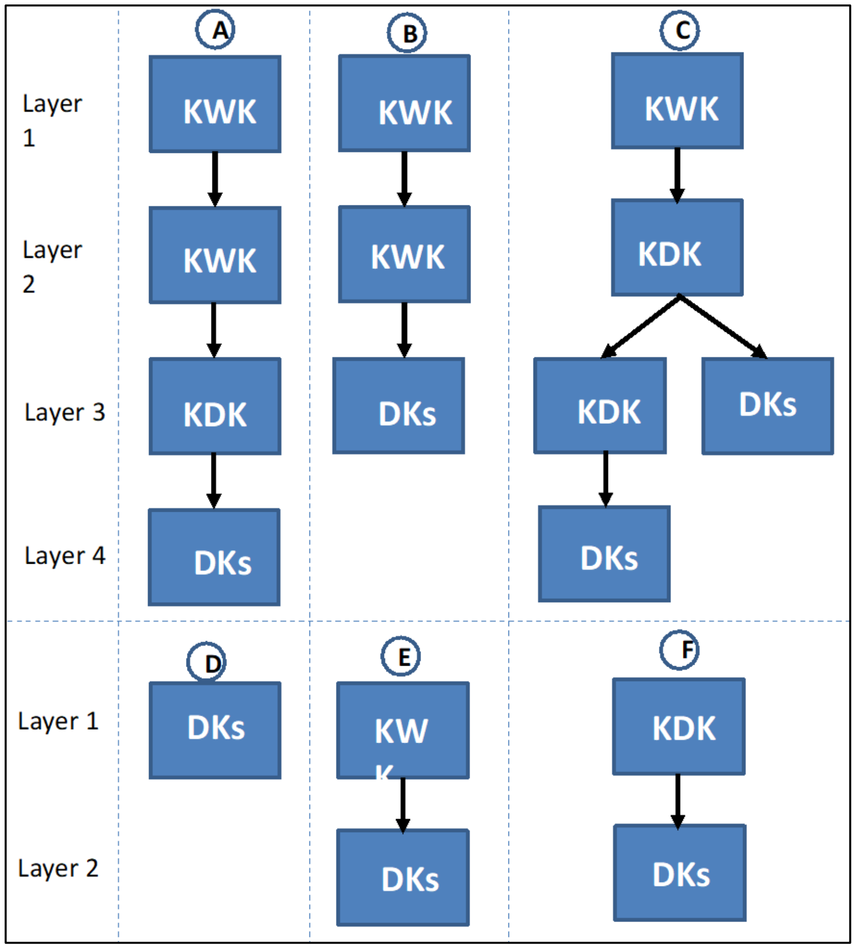

Figure 1: Examples of Symmetric-Key Hierarchies

Figure 1 provides several examples of symmetric-key hierarchies.

- The top-most layer (Layer 1) can be any of the key types. This layer establishes a keying relationship.
- When the Layer 1 key is a KWK, further keys may be distributed using that KWK (see examples A, B, C and E in which KWKs, KDKs, and DKs are shown at Layer 2 in the figure).
- A KDK at any layer has the data keys and KDKs that it derives as a lower layer (see examples A, C and F).
- DKs (i.e., DEKs, DAKs and AEKs) are always at the bottom of the implemented hierarchy, even if the DK is a Layer 1 key, in which DKs form the only layer in the hierarchy (see example D).

- KWKs, KDKs and DKs in a layer immediately below KWKs are wrapped by the KWK above them in the hierarchy (see examples A, B, C and E).

The key hierarchy may not be "vertical" as shown in Figure 1 but may be somewhat more horizontal; two examples are shown in Figure 2.

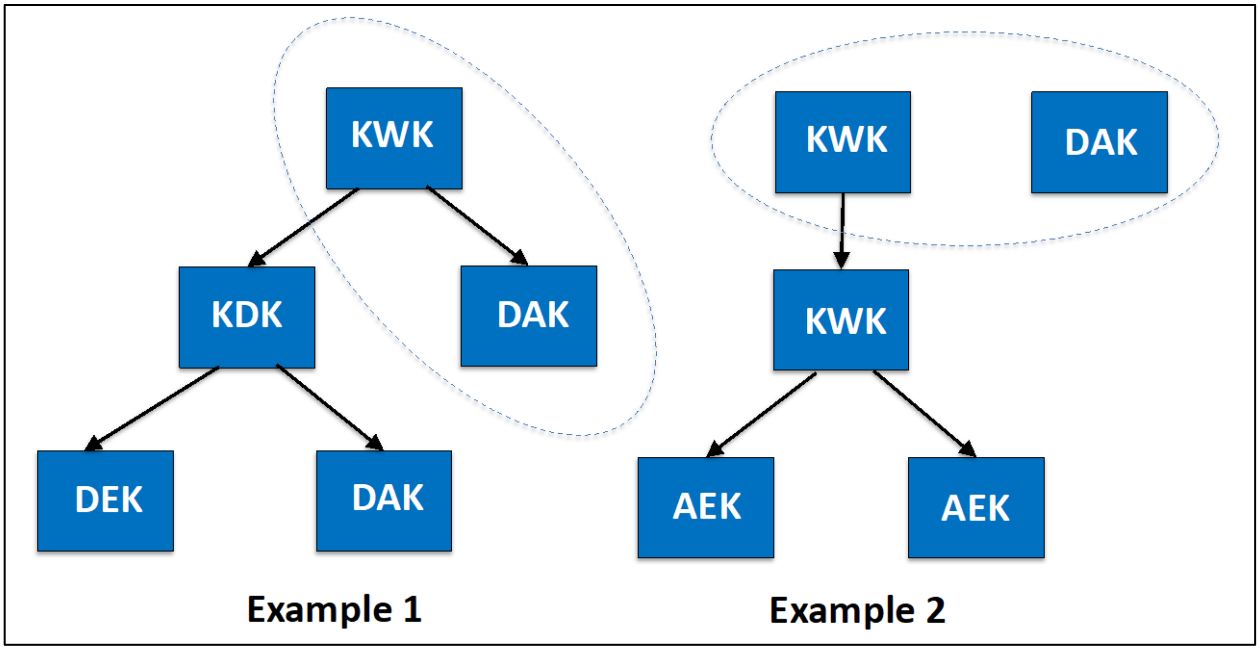

Figure 2: Key Hierarchy Structure Examples

In example 1 of the figure, the Layer 1 KWK was used to wrap a Layer 2 DAK; these keys were used to establish a keying relationship (indicated in the left-hand oval). Subsequently, the KWK was used to wrap a Layer 2 KDK, which was used to generate a Layer 3 DEK and DAK.

In example 2 of the figure, the KWK and DAK established the keying relationship (indicated in the right-hand oval), but the DAK was not wrapped using the KWK as was done in the first example. In this case, both the KWK and DAK are Layer 1 keys. Subsequently, the KWK was used to wrap a Layer 2 KWK, which was later used to wrap two Layer 3 AEKs.

For the most part, the number of layers is irrelevant; the important issue is where the key is located in a hierarchy, especially if the revocation of a key is required (see Section 5.5).

3.5.1 Storage Applications

All keys used to protect stored target data **shall** be either generated by the system in which the target data is stored or generated by the sender of cryptographically protected data that is stored by the recipient upon receipt. As stated in Section 3.5, the lowest layer in the key hierarchy consists of the data keys (i.e., DEKs, DAKs and AEKs) used to protect the stored target data. Higher-layers of keys, if used, are the KWKs used to protect the data keys or the KDKs used to derive them (see Figure 1 and Figure 2).

3.5.2 Communicating Groups

The use of symmetric keys for communications between correspondents requires the establishment of cryptographic keying relationships among two or more entities that form a communicating group (i.e., a group of entities that correspond among themselves); often, a communicating group consists of only two entities. An entity may be a member of more than one communicating group.

When using symmetric-key cryptography, a keying relationship is established when each member of the group shares common keys – the Layer 1 keys of that relationship. Symmetric keying relationships among communicating groups are established using the methods in Section 3.4 or using key centers (see Section 4.1). Section 4.2 provides more details regarding the establishment of communicating groups.

The keys used during communications among communicating group members (either the Layer 1 keys or keys below them in a key hierarchy) may be either uni-directional or bi-directional.

- Uni-directional keys are used in only one direction during communications among group members. Each group member that is authorized to send data has its own key for applying cryptographic protection (e.g., encrypting data) to be sent to other group members. Other members of the group have copies of the keys, but only use them for processing (e.g., decrypting) the cryptographically protected information. For example, if Entities A and B are the members of a communicating group, Entity A would use a key for encryption, but Entity B would use that key only for the decrytion of information from Entity A. Entity B would use a different key for encryption, and Entity A would use that same key only for the decryption of information from Entity B. This approach is most appropriate for very small groups (e.g., communicating pairs), or when very few group members are authorized to apply protection.
- Bi-directional keys can be used in both directions during a communication between group members; the same symmetric key is used by each member for both protecting (e.g., encrypting) sensitive data to be sent to other group members and for processing (e.g., decrypting) protected data received from other group members.

3.5.3 Key-Establishment Transactions

A key-establishment transaction is an instance of establishing keying material among or between entities. This includes requests for generating keys, the generation of the keys, the distribution of those keys and a confirmation of delivery. This applies to both manual and automated key distribution.

For automated key distribution, this requires multiple protocol messages. The integrity of each message and assurance of the message source is provided using a message authentication code (MAC) that is generated using a transaction authentication key generated for the transaction or a DAK shared between the message sender and receiver when a transaction authentication key is not available (e.g., in error messages in response to messages containing the transaction authentication key).

4. Key Management Architectures for Symmetric Keys

This section describes architectural considerations for the establishment of symmetric keys and specifies architectures for different key-establishment environments. Because the security of cryptographically protected systems is largely dependent on the effectiveness of key management architectures, any such architecture must take into account organizational structures and responsibilities, and operational requirements. Key-management architecture design is best undertaken by specialists who have a comprehensive understanding of the organization, its requirements, and the risks to which it is exposed. This section describes architectural elements in general and some of the considerations associated with the design, selection, and acceptance of key management architectures.

This section provides high-level examples of key-establishment using symmetric-key systems. The general architectural approaches described include center-based key establishment and key establishment for communicating groups. Section 5 provides further information on the messages used for key establishment, and Appendix A provides more in-depth examples.

4.1 Center-based Key Establishment Architectures

Key centers can be used to mitigate one of the primary objections to the use of symmetric keys for cryptographic protections: the number of keys required to initiate and maintain cryptographic keying relationships between communicating entities (i.e., members of communicating groups) when asymmetric keys are not available for this purpose. When using key centers, each entity becomes a subscriber of a mutually trusted key center by establishing a cryptographic keying relationship with that center consisting of a KWK and a DAK. The KWK is used to wrap keying material for transport, and the DAK is used to authenticate messages when another authentication key is not available. A KWK and DAK shared between any subscribing entity and a center permits secure communications to be established between that entity and any other subscribing entity that has a KWK shared with the center.

A keying relationship between a center and its subscribers is normally established using a manual process whereby either the center or the subscriber generates the keying material and provides it to the other party. The relationship is rekeyed using the same process. Alternatively, if an asymmetric key-establishment capability is available (e.g., asymmetric key agreement or key transport), the keying material could be established using that capability. See Section 3.4.

For center-based key establishment, the center is responsible for verifying the identity of each of its subscribers, authorizing communications between subscribers by providing or not providing the services of the center, and may provide secure key-generation services.

Key center architectures have several variants: Key Distribution Centers (KDCs), Key Translation Centers (KTCs) and Multiple-Center Groups of KDCs and/or KTCs. Figure 3 depicts the keying relationships between a single center and its subscribers. The center may be either a KDC or KTC. As shown in the figure, each subscriber shares a different KWK with its center.

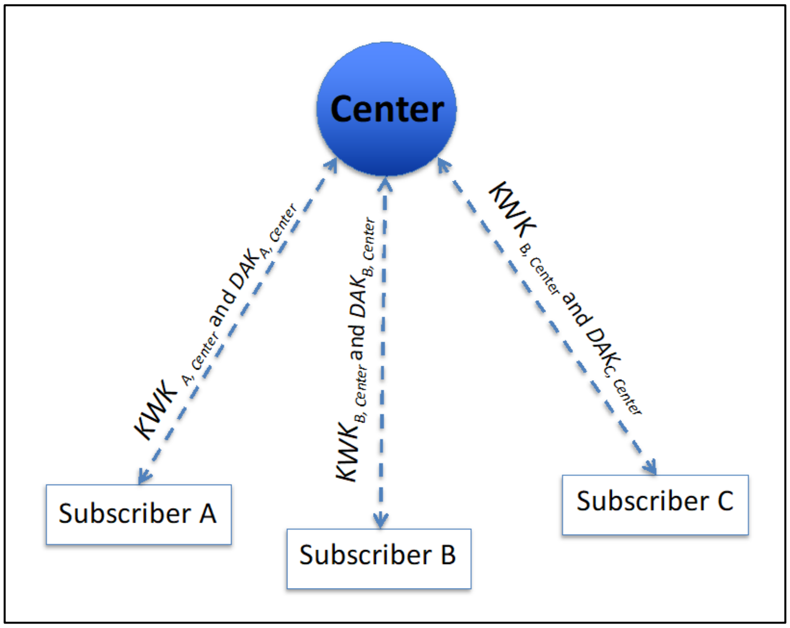

Figure 3: Center-Subscriber Keying Relationships

The keying relationship between a subscriber and a center can be used to establish keying relationships between non-center entities (e.g., subscribers A, B and C in the figure) to form communicating groups of two or more entities using automated key-establishment protocols. In cases where a KWK and DAK are established as the Layer 1 keys among subscribing entities, and at least one of those entities has key generation capabilities, subsequent key-establishment transactions may be performed without using the key center (see Section 4.2.2). The KWK and DAK that are established using the services of a key center **shall** only be replaced using the services of that center.

4.1.1 Key Distribution Centers (KDCs)

A KDC is responsible for the secure generation and distribution of keys to its subscribers, either to be used by a single subscriber for its own purposes or to be shared by multiple subscribers. KDCs may send keys either unsolicited or upon request.

When keys are intended to be shared by multiple subscribers, the KDC generates and distributes keys to subscribing entities who:

- Need to communicate with each other but either 1) do not currently share keys, 2) need to replace keys previously established using that KDC or 3) the KDC determines (of its own volition) that keys need to be shared between a subset of subscriber entities that will form a communicating group;

- Each share a KWK and DAK with the same KDC (i.e., each entity is a subscriber of the same KDC); and
- May not have the ability to generate keys.

A copy of the keys for each identified subscribing entity is wrapped by the KDC using a KWK shared between that entity and the KDC. The wrapped keys may be sent to one subscribing entity (e.g., the requesting entity) to be forwarded to the other entity(ies) (see Figure 4), or may be sent directly to the (recipient) entities (including the requesting entity), depending on the protocol (see Figure 5).

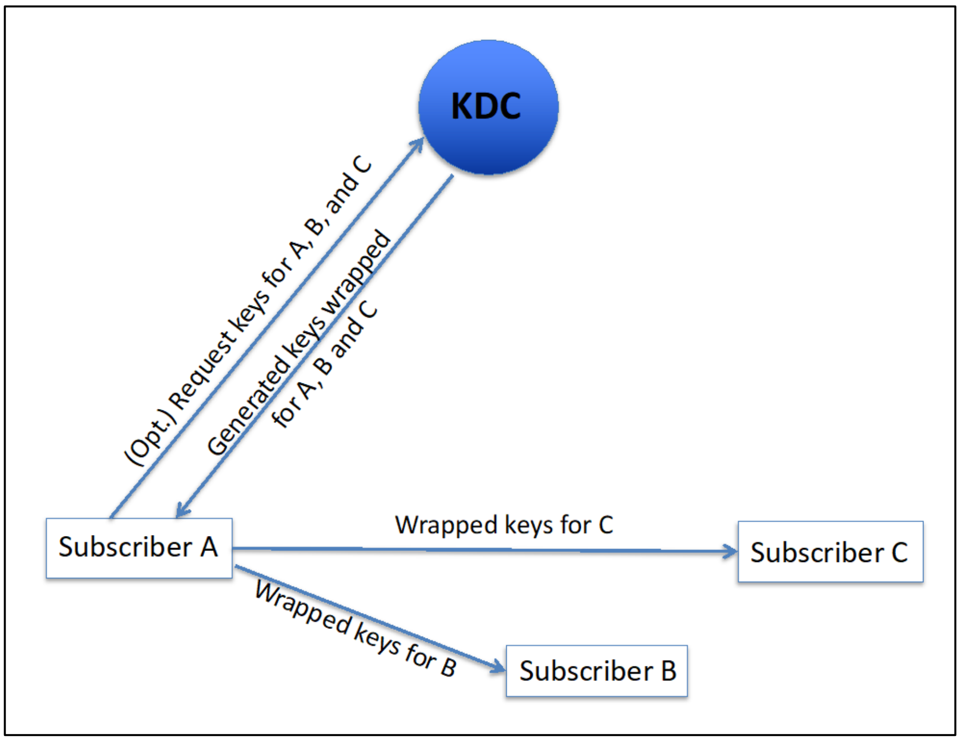

Figure 4: Obtaining Keys from a KDC (Distributing through a Single Subscriber)

Using Figure 4 as an example:

1) Subscriber A may optionally request that the KDC generate keying material, indicating other subscribers that need to share the key (i.e., Subscribers B and C in the figure); the DAK shared between Subscriber A and the KDC is used for message authentication.

2) Alternatively, the KDC may initiate the key distribution process without a subscriber request by generating keying material to be shared by some subset of its subscribers (e.g., Subscribers A, B and C in the figure).

3) In either case, the KDC generates the requested keying material, wraps it separately using the KWK shared with each subscriber intended as a recipient. A transaction authentication key (i.e., DAK) is also generated and wrapped; this DAK is in addition to any other DAK included in the requested keying material.

4) In this example, the KDC sends all wrapped copies of the keys to Subscriber A in a message that uses the transaction authentication key to generate a MAC on the outgoing protocol message.

5) Subscriber A extracts its copy of the keys from the message and unwraps them using the KWK shared with the KDC. The unwrapped transaction authentication key and the received authentication code are used to check the authenticity of the message.

6) If the message appears to be authentic, subscriber A forwards the appropriate copy of the keying material to the other intended recipient subscribers (i.e., Subscribers B and C in this example) using the transaction authentication key to generate a (different) MAC on each outgoing message.

7) Each recipient unwraps the received keying material using the KWK that is shares with the KDC and uses the unwrapped transaction authentication key to check the authenticity of the message.

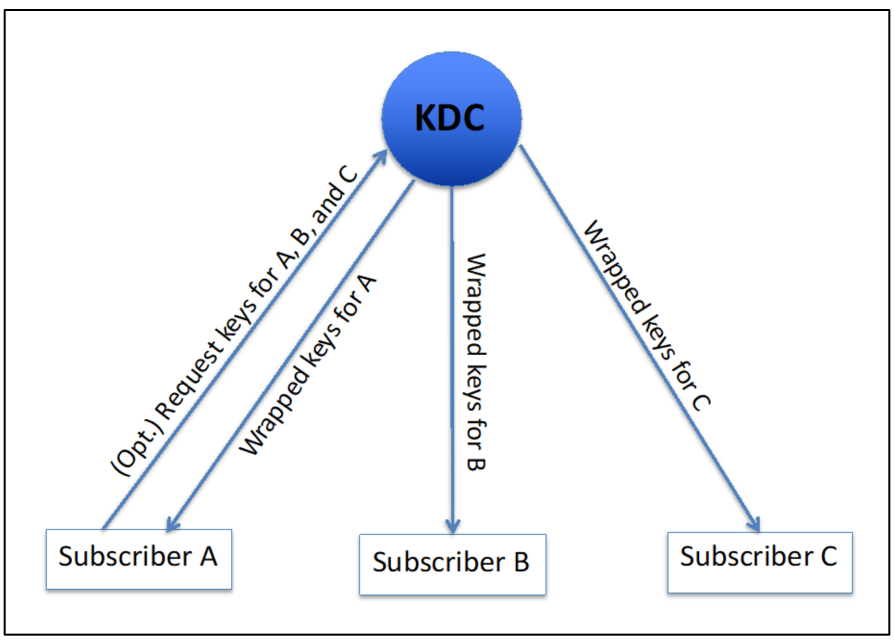

Figure 5: Obtaining Keys from a KDC (KDC Distributes Keys to Each Subscriber Separately)

Using Figure 5 as an example: steps 1, 2 and 3 are the same as the example above.

4) The KDC sends a message to each intended recipient (including Subscriber A) containing the appropriate copy of the wrapped keying material and using the unwrapped transaction authentication key to generate a (different) MAC on each outgoing message.

5) Each recipient unwraps the received keying material using the KWK that it shares with the KDC and uses the unwrapped transaction authentication key to check the authenticity of the message.

The scenario described in Figure 5 places more responsibility for key management overhead (e.g., accounting, revocation and suspension notice, etc.) on the KDC, while that described in Figure 4 places more overhead responsibility on Subscriber A. Organizational structures and assignments of responsibilities can play a significant role in deciding which approach is preferable.

4.1.2 Key Translation Centers (KTCs)

A Key Translation Center has the ability to translate keys for distribution to a subset of its subscribers.

A KTC is used to translate keys for future communication between subscriber entities of the same KTC who:

- Need to communicate with each other, but may not currently share keys;
- Each share a KWK and DAK with the same KTC (i.e., each entity is a subscriber of the same KTC); and
- At least one of the subscribing entities has the ability to generate keys.

Keying material is generated and sent by one of the subscribers (the requesting entity) to the KTC, wrapped using the KWK shared with the KTC. The KTC unwraps the keying material to be translated and rewraps it using the KWK shared with other identified subscribing entity(ies) (i.e., the ultimate recipient(s)). The rewrapped keying material may be returned to the requesting entity to be forwarded to the ultimate recipient(s) (see Figure 6), or may be sent directly to the ultimate recipient(s), depending on the protocol (see Figure 7).

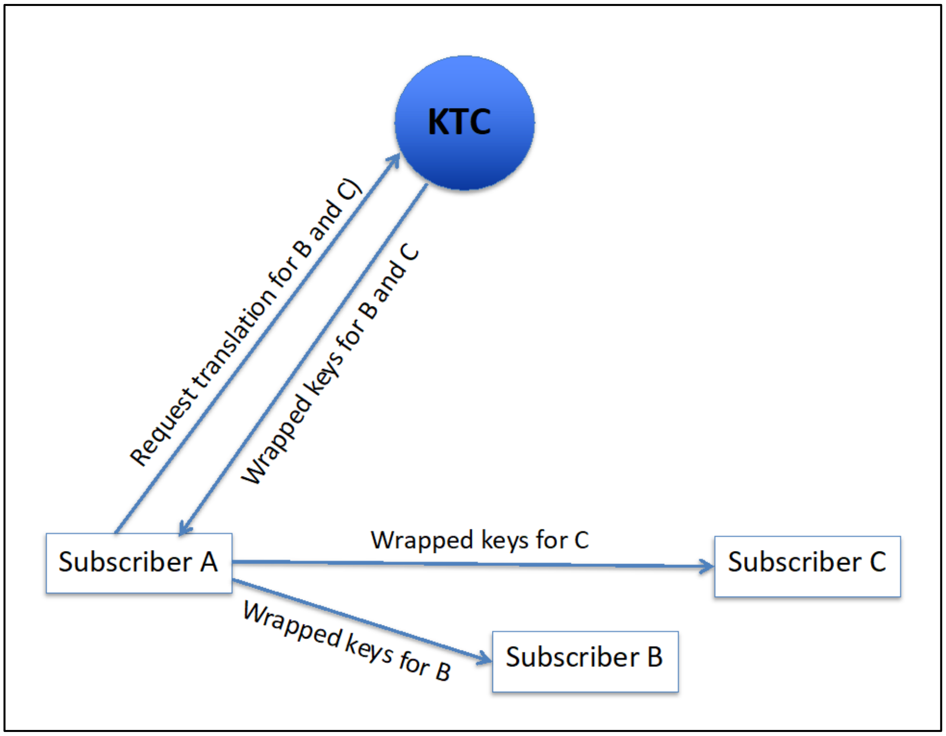

Figure 6: Requesting Key Translation, with Keys Provided through a Single Subscriber

Using Figure 6 as an example:

1) Subscriber A generates keying material to be shared among other KTC subscribers (i.e., Subscribers B and C in the figure); a transaction authentication key (i.e., DAK) is also generated.

2) Subscriber A wraps the keying material (including the transaction authentication key) using a KWK shared with the KTC and sends it to the KTC, indicating other subscribers that need to share the keys (i.e., Subscribers B and C); the transaction authentication key is used to generate a MAC on the outgoing message.

3) The KTC unwraps the received keying material using the KWK shared with Subscriber A and uses the unwrapped transaction authentication key to check the authenticity of the received message.

4) If the received message appears to be authentic, the KTC then rewraps the keying material separately for each intended recipient using the KWK shared with that recipient.

5) The KTC prepares a message containing the newly wrapped keys, generates a MAC on the message using the (plaintext) transaction authentication key, and sends the message to Subscriber A.

6) Subscriber A forwards the appropriate copy of the keying material to the other intended recipient subscribers (i.e., Subscribers B and C) using the transaction authentication key to generate a (different) MAC on each outgoing message.

7) Each recipient unwraps the received keying material using the KWK that is shares with the KTC and uses the unwrapped transaction authentication key to check the authenticity of the message.

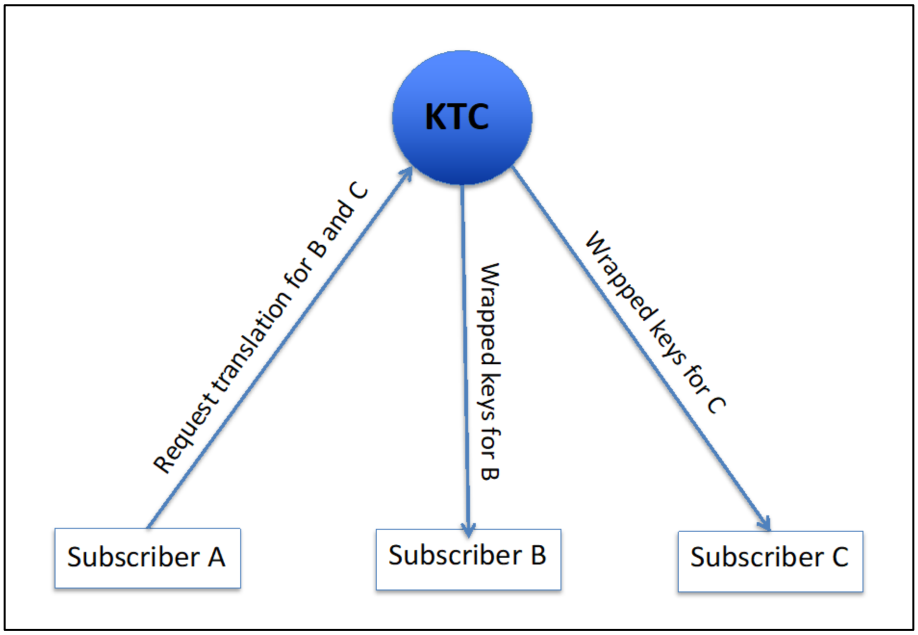

Figure 7: Requesting Key Translation, with Keys Returned Separately to Each Subscriber

Using Figure 7 as an example, steps 1 through 4 are the same as the above example.

5) The KTC sends a message to each intended recipient (B and C) containing the appropriate copy of the wrapped keying material, using the transaction authentication key to generate a (different) MAC on each outgoing message.

6) Each recipient unwraps the received keying material using the KWK that it shares with the KTC and uses the unwrapped transaction authentication key to check the authenticity of the message.

As in the case of KDCs (see Section 4.1.1), organizational structures and assignments of responsibilities can play a significant role in deciding which approach is preferable.

4.1.3 Multiple-Center Architectures

A multiple-center group is a set of two or more centers (KDCs and/or KTCs) that have formally agreed to work together to provide cryptographic keying services to their respective subscribers. To the subscribers of a key center, the multiple-center group functions as if it were a single key center. Key centers may belong to more than one multiple-center group, but care **shall** be taken to separate domains of subscribers (e.g., subscribers for one organization from subscribers of another organization).

Each center within the group has a keying relationship with at least one other center in the group: the centers share a KWK and a DAK to transport keying material between them. The centers may also distribute other keying material using their shared keys to protect messages exchanged between the centers.

Every center within a multiple-center group **shall** have either a direct or an indirect keying relationship with every other center within the group (see Figure 8). Two centers have a direct keying relationship when they share a KWK and DAK (established as discussed in Section 3.4). Once the multiple-center group is established, the multiple center group **shall** use either manual or automated protocols to maintain these keying relationships (i.e., to change the shared key(s)).

Two centers have an indirect keying relationship when they do not share a KWK and DAK, but there is a chain of direct keying relationships between them. In Figure 8, for example, direct keying relationships exist between Centers 1 and 2, and between Centers 2 and 3. An indirect keying relationship exists between Centers 1 and 3 because of the direct relationships that form a chain of keying relationships through Center 2.

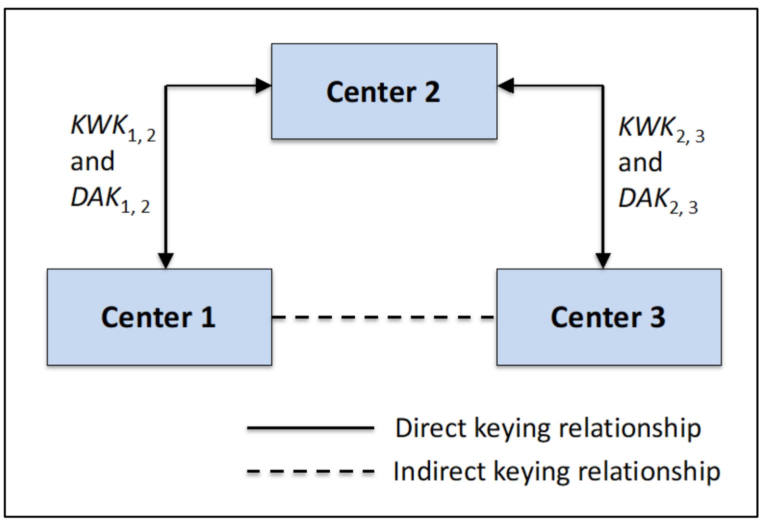

Figure 8: Multiple-Center Group Direct and Indirect Keying Relationships

The use of indirect keying relationships can reduce the key management overhead associated with deploying keys among the multiple-center group members but can also reduce central control over relationships in a hierarchical environment.

Centers within multiple center groups may provide key generation services. All centers within the group that have subscribers **shall** be capable of providing key translation; only a subscriber's agent (i.e., a key center to which an entity is subscribed) **shall** translate (i.e., wrap) keys for that subscriber. Some centers within the multiple-center group may only forward the keys or a request for services to the next center.

Intermediate centers within the multiple center group forward information when a direct keying relationship does not exist between the agents or between an agent and a center that will generate the key(s) or perform the translation. The intermediate centers used in one portion of the information flow need not be the same as those used in another portion of the information flow. However, the number of intermediate centers used **should** be minimized.

A multiple-center group **shall** be well-defined; all centers within a multiple-center group must be aware of what other centers are members of the group as well as the conditions and restrictions for group interactions. If a center belongs to more than one group, the interactions of one group **shall** be separated from the interactions of another group.

The centers within a multiple-center group have specific keying relationships between them and use communication protocols to manage those keying relationships[40] and fulfill requests from their subscribers.

Multiple-center groups can be used to support the establishment of keying relationships between subscribers of different centers that belong to the multiple-center group (e.g., to establish communicating groups). Depending on the group design, every subscriber of a center within the group may or may not be able to establish keys with all subscribers of all centers within the group.

Each subscribing entity associated with a KDC or KTC within a multiple-center group has a keying relationship with at least one center that is a member of the group; this center is the subscriber's "agent" for the group; however, a center need not have subscribers of its own. Entities (i.e., subscribers) may have more than one agent for a multiple-center group, and a subscriber may subscribe to more than one multiple-center group using the same or a different agent. Using a center as an agent to a group does not preclude using the same center as a single-center KDC or KTC. Interaction with the other members of a multiple-center group by an agent is a service provided by that agent center. Key transactions initiated by a subscriber to one of its agents **shall** be fulfilled or acknowledged to the subscriber through that same agent.

The following services may be provided to the subscribers by a multiple-center group.

- A key-distribution service is equivalent to the service provided by a single-center KDC. One or more centers within the group **shall** be capable of generating keying material; however, only one center (i.e., only one KDC) within the multiple-center group **shall** generate the keying material for a single key-distribution process. Key generation may be in response to a request from a subscriber to its agent (one of the centers within the multiple-center group) or as determined by a center within the group. Agent centers within the multiple-center group

[40] This is similar in concept to the use of cross certification between PKI Certification Authorities.

are responsible for wrapping the key(s) intended for their subscriber(s) under KWKs shared with those subscribers. All copies of the wrapped keys may be sent to a single subscriber for forwarding to the other intended recipients or provided to each recipient subscriber by its multiple-center agent.

- A key-translation service is equivalent to the service provided by a single-center KTC. KTCs forward copies of the key to the appropriate center(s) within the group for each subscriber designated to receive a copy of the key. All copies of the wrapped keys may be sent to a single subscriber or provided to the subscribers by their respective multiple-center agent.

The following subsections provide high-level examples of interactions between subscribers and their agent centers and between centers within the multiple-center group. In these examples, all keying material is sent directly to the intended recipient(s). However, the keying material could also be returned to one subscriber, who distributes it to other subscribers. The examples do not address error handling; this is included in the more-detailed examples in Appendices A.5 and A.6.

4.1.3.1 A Subscriber Requests Key Generation and Distribution Services

In this example, a subscriber of an agent center requests key-generation services for keying material to be subsequently shared among a list of entities that are subscribers of some agent within the multiple-center group; this process starts at step 1) below. Alternatively, a center within the group could initiate the process for a predetermined list of entities (starting at step 2a or 3a).

1) A subscriber (i.e., the requesting subscriber) sends a key-generation request to its agent center, indicating the other intended recipients of the keying material. A MAC is generated on the outgoing message using the DAK shared with the agent.

2) If the agent can generate the requested keying material:

   a) The agent generates the requested keying material and a transaction authentication key, wraps a copy of the keys (including the transaction authentication key) for the requesting subscriber using the KWK shared with that subscriber, and sends them to the subscriber, using the transaction authentication key to generate a MAC on the outgoing message.

   b) For each intended recipient that is a subscriber of that agent: The agent wraps a copy of the keys using the KWK shared with each intended recipient and sends them to that subscriber, using the transaction authentication key to generate a MAC on the outgoing message. Alternatively, these copies could be sent in the same message as those intended for the requesting subscriber (see step 2a).

   c) For each intended recipient that is not a subscriber of that agent:

      - The agent attempts to determine a path through the multiple-center group to that recipient's agent for translation of the keys to be sent to that recipient.

    - The keying material is wrapped using a KWK shared with the next center in the path for transport to that center. A separate transaction authentication key is generated for multiple-center group communications, wrapped using the KWK shared with the next center in the path, and used to generate a MAC on the outgoing message.

  d) If a center receives a translation request from another center within the multiple center group, and that center is not the agent for the intended recipient (i.e., the center is an intermediate center):

    - The receiving center unwraps the received keying material using the KWK shared with the previous center and uses the unwrapped group transaction authentication key to check the authenticity of the received message.
    - If the received message appears to be authentic, the center then attempts to determine a path to the intended recipient's agent and wraps the keying material (including the group transaction authentication key) using a KWK shared with the next center in the path for transport to that center.
    - A MAC is generated on the outgoing message using the unwrapped group transaction authentication key.

  e) If a center receives a translation request from another center within the multiple center group, and the receiving center is the agent for the intended recipient:

    - The agent center unwraps the received keying material using the KWK shared with the previous center and uses the unwrapped group transaction authentication key to check the authenticity of the received message.
    - If the received message appears to be authentic, the agent center wraps the keying material to be translated (including the transaction authentication key, but not the group transaction authentication key) and sends it to the intended recipient, using the transaction authentication key to generate a MAC on the outgoing message.

  f) Any subscriber receiving wrapped keying material unwraps it using the KWK shared with its agent and uses the unwrapped transaction authentication key to check the authenticity of the received message.

3) If the agent center or another center within the group receives a request to generate keying material but is unable to do so:

  a) The center checks the authenticity of the received message using the DAK shared with the subscriber or center having sent the generation request.

     The center then forwards the request to a center within the group that can generate the requested keying material. Intermediate centers may be required. The forwarded request uses the DAK shared with the next center to generate a MAC on the outgoing message.

b) When a center with a key-generation capability (a key-generation center) receives the request:

- The authenticity of the received message is checked using the DAK shared with the subscriber or center that sent or forwarded the generation request.
- The requested keying material is generated, as well as a transaction authentication key.
- Keying material destined for any subscriber of that center is wrapped using a KWK shared with that subscriber and sent to that subscriber, using the transaction authentication key to generate a MAC on the outgoing message.
- For any intended recipient that is not a subscriber of the key-generation center, go to step 2c above, proceeding through steps 2 d, e and f, as appropriate.

4.1.3.2 A Subscriber Requests Key-Translation Services

In this example, a subscriber of an agent center generates keying material and requests translation services for keying material to be subsequently shared among a list of entities that are subscribers of some agent within the multiple-center group.

1) A subscriber generates keying material (including a transaction authentication key), wraps the keys using a KWK shared with its agent center and sends a translation request to the agent, indicating the intended recipients. The transaction authentication key is used to generate a MAC on the outgoing message.

2) When an agent center receives a translation request:

   - The center unwraps the keying material using the KWK shared with the requesting subscriber and checks the authenticity of the received request using the unwrapped transaction authentication key.

   - If any intended recipient is also a subscriber of that agent center, the agent wraps the keying material (including the transaction authentication key) using a KWK shared with that recipient and sends it to the recipient, using the transaction authentication key to generate a MAC on the outgoing message.

   - For any intended recipient that is not a subscriber of that agent:

     - The agent attempts to determine a path through the multiple-center group to that recipient's agent for translation of the keying material.

     - If the agent is capable of generating keys, a group transaction authentication key is generated.

     - The keying material is wrapped (including the newly generated group transaction authentication key, if available) using a KWK shared with the next center in the path for transport in a translation request to that center. A MAC is generated on the outgoing message using the newly generated group transaction authentication key (if available) or using the transaction authentication key sent in the translation key if not; an indication of which key is used must be present (referred to as the "appropriate authentication key" below).

3) If a center receives a translation request from another center within the multiple center group, and that center is not the agent for the intended recipient (i.e., the center is an intermediate center):

   - The receiving center unwraps the received keying material using the KWK shared with the previous center and checks the authenticity of the received message using the "appropriate authentication key."
   - The center then attempts to determine a path to the intended recipient's agent and wraps the keying material (including the "appropriate authentication key") using a KWK shared with the next center in the path for transport to that center.
   - A MAC is generated on the outgoing message using the "appropriate authentication key."

4) If a center receives a translation request from another center within the multiple center group, and that center is the agent of the intended recipient:

   - The agent center unwraps the received keying material using the KWK shared with the previous center and uses the unwrapped "appropriate authentication key" to check the authenticity of the received message.
   - The agent center wraps the keying material to be translated (including the transaction authentication key, but not the group transaction authentication key, if present) and sends it to the intended recipient, using the transaction authentication key to generate a MAC on the outgoing message.

5) Any subscriber receiving wrapped keying material unwraps it using the KWK shared with its agent and uses the unwrapped transaction authentication key to check the authenticity of the received message.

4.2 Communicating Groups

This section discusses the establishment of a communicating group and the subsequent distribution of additional keying material within that group. Also see Section 3.5.2.

4.2.1 Establishing Communicating Groups

Communicating groups of two or more entities share keys for communication among the group members. Prior to or during the establishment of a communicating group, each prospective member of the group **shall** have assurance of the validity of the group, the source and method of group establishment, the identity of the other group members and the rules for group operation (e.g., using contracts, security policies, memoranda of agreement).

The Layer 1 keys that establish the keying relationship among the group members **shall** be established using one of the following methods:

a) Manual distribution as discussed in Section 3.4.1,

b) Use key centers or multiple-center groups: Each member of a prospective communicating group would become a subscriber of the same key center or of an agent center of the same multiple-center group (see Section 3.4 and Section 4.1).

  - One member of the intended group requests the generation of a key for distribution to the other intended members of the group from a KDC or multiple center group (see Sections 4.1.1 and 4.1.3),
  - A KDC or multiple-center group (of its own volition) generates a key and sends it to the intended group members (see Sections 4.1.1 and 4.1.3),
  - One member of the intended communicating group generates a key and sends it to a KTC for translation for the intended members of the group (see Section 4.1.2), or

c) The key is established using asymmetric key-establishment methods. Each member of a prospective communicating group could obtain an asymmetric key-establishment key pair and the associated public key certificate. The entity generating the Layer 1 keys could then use the public key associated with each group member to distribute the Layer 1 keys to that member.

4.2.2 Communicating Group Requirements

1) A key used by any communicating group **shall** not intentionally be used by any other communicating group.

2) When replacing a Layer 1 key (e.g., at the end of its cryptoperiod or because of a compromise), the key **shall** be replaced in the same manner as it was established.

3) A key shared among a communicating group **shall** not be disclosed to a different communicating group. If a member of a communicating group is also a member of another group, that member **shall not** use or disclose that key to other members of the other group. For example, if entity A is a member of both group 1 and group 2, a key used in group 1 **shall not** be either used or disclosed to other members of group 2 unless they are also members of group 1.

4) A key shared among a communicating group **shall** be secured from third entity usage, except for an entity that was involved in the distribution of that key (e.g., a key center, see Section 4.1).

5) A key that has been used by a communicating group or other cryptographic keying relationship and revoked **shall** not be intentionally re-used for any subsequent interaction (also see Section 5.5).

4.2.3 Subsequent Key Distribution within a Communicating Group

Once a communicating group is established (see Section 4.2.1), the members can operate as peers or in master/recipient relationships for subsequent key-distribution operations. A peer relationship exists when all members of the group are allowed to generate or otherwise obtain (e.g., using a KDC) keying material for distribution to the other members of the group. In a master/recipient relationship, one (or more) members of the group (i.e., masters) are allowed to generate keying material and distribute it to all other members of the group, while other members (i.e., recipients) are only allowed to receive keying material.

Note that keys can only be distributed using automated methods if the group shares a KWK.

When the group shares a KDK (e.g., examples A, C and F in Figure 1), some member (or pre-established rule) needs to decide when to derive a new data key or KDK from the already-shared KDK.

If a communicating group shares a KWK (e.g., examples A, B, C and E in Figure 1), and at least one member of the group has a key generation capability, then additional keys may be generated and distributed within the group without the assistance of key centers. The member entity that generates the key wraps the newly generated key under a KWK shared with the other members of the group (the recipients); the recipients unwrap the received key using the same shared KWK.

5. Key-Establishment Communications

This section addresses key-establishment communication requirements for communicating groups; communications between KDCs, KTCs and multiple-center groups and their subscribers; and communications among the member centers in a multiple-center group.

5.1 General Communications Requirements

Automated symmetric-key establishment is dependent on communications among components of the key-management infrastructure. Communications in support of key establishment **should** be accomplished according to system-wide protocols. The protocols **shall** establish the key-establishment information content to be used for the automated establishment of keying material.

Although this Recommendation does not specify key-management protocols, some general guidelines are offered regarding processing rules to be followed in key establishment.

a) All key-establishment messages **shall** have integrity and source authentication protection using an **approved** message authentication algorithm (e.g., HMAC or KMAC) and a secret DAK shared between the sender and receiver.

b) When a key-generation capability is available, a newly generated authentication key **should** be generated for an outgoing message containing keys if a KWK is available for wrapping the authentication key.

c) Messages carrying keys **shall** include the key(s) used to authenticate the message, protected by a shared KWK or a KWK sent in the message.

d) Before taking action on received key-establishment messages, the receiving entity **shall**:

- Attempt to verify the authentication code in the received message (see Section 5.4). If an error is detected, an error message **shall** be sent to the message sender.
- Check the authenticity/validity/authorizations/reasonableness of other information carried in the received message (e.g., using nonces or sender IDs). If an error is detected, an error message **shall** be sent to the message sender.
- If another message is not to be immediately sent to the message sender in response to a request, an acknowledgement message **shall** be sent to the message sender.

e) Messages sent in response to messages carrying keys **should** use the authentication key sent in the previous message for computing the authentication code on the responding message. However, if the authentication code in the received message could not be verified, another authentication key shared by the message sender and receiver **shall** be used for computing the message authentication code on the responding error message.

f) In order to facilitate secure key establishment, keys may need to be uniquely identified by key names or labels. The assignment of a label to a key **shall** be made only by the entity or center that generates the key or requests the generation of a key with a specified label. Once a key is labeled, the label **shall** not be changed.

Keys may be uniquely identified by:

(1) The sharing entities,

(2) A key identifier (i.e., key name),

(3) The key type (e.g., KWK or DEK),

(4) The key subtype (i.e., manually or electronically distributed KWK, authentication or encryption data key) and/or

(5) The effective date of the key[41].

The sharing entities and key type of a key carried in a key-establishment message or used to wrap a key carried in a key-establishment message **should** be specified in that message.

5.2 Notation

In Section 5.3 and the examples in Appendix A, information sets to be included in protocol messages are formatted as follows:

information_name(*parameter_set_1*; *parameter_set 2* {; ...; *parameter_set_n*}),

where each parameter set has one or more items separated by commas.

Keys and variable information in a parameter set is italicized; other information is not.

Keys are indicated by the type of key (e.g., KWK), subscripted by the identities of the sharing entities or the name of the key, e.g., $KWK_{A,\,B}$ or DAK_{name}.

Keys are wrapped as follows:

wrapped_keys = WRAP(*wrapping_key*, *concatenation_of_keys_to_be_wrapped*).

5.3 Message Content and Handling

This section recommends information that **should** be included in key-establishment messages and provides guidance in handling them when received. The section identifies information that needs to be included in each type of message, but not the format to be employed or other information to be included in a specific protocol message.

Key-establishment messages are used to:

- Request the generation or translation of keying material,
- Acknowledge the receipt of a request for key generation or translation services,
- Provide keying material to other entities,
- Acknowledge the receipt of keying material,
- Request the revocation of keys,
- Confirm that keys have been destroyed in response to a revocation request, and
- Report errors in key-establishment messages, providing information that may allow error recovery.

[41] The effective date could be the begin date, the end date, or the cryptoperiod.

Additional message types may be required by a particular key-management infrastructure.

The following key-establishment information sets are recommended for incorporation into protocol messages. Specific protocols may combine the functionality of two or more sets of recommended information into a single protocol message when appropriate. Similarly, a protocol may employ more than one protocol message to convey the information identified for each information set listed below. The specification of the actual messages to be used in a communication protocol **should** be carefully designed to fulfill the participant's policy requirements for secure communications.

Examples for using these information types are provided in Appendix A.

5.3.1 Key Generation Request

A *key generation request* permits entities within a key-management infrastructure to request that keys be generated. The request may be sent:

- By one member of a communicating group (entity B) to another member of that group (entity A),
- From one entity (B) to another entity (A) to request the services of a KDC, KTC or multiple-center group (e.g., to establish a communicating group),
- By a KDC subscriber (A) to a KDC or a multiple-center group, or
- By a center within a multiple-center group (Center 1) to another center within the group (Center 2) who may or may not have a key-generation capability.

The request may be transitive (e.g., from Subscriber B through Subscriber A to a KDC), and the request may be forwarded until it is received by an entity that is capable of servicing the request (e.g., from B to A to Center 1 to Center 2) or until it is determined that the service is not available.

A *key-generation request* **shall** indicate the types of keying material to be generated (*requested_keys*) and the intended communicating group [42] for using those keys. The *requested_keys* information might include the key types (e.g., KWK and DAK), the algorithm (e.g., AES), the key length (e.g., 256 bits) and a label for each key, or other information, as appropriate.

The request **shall** provide for the authentication of the requesting entity and for integrity protection of the message containing the *key-generation request* information using an authentication key (*auth_key*) to generate an authentication code (*auth_code*) on the message (see Section 5.4):

key-generation_request(*requested_keys*; *communicating_group*; *auth_code*).

See example scenarios in Appendices A.1, A.2, A.3 and A.5 for the use of *key-generation request* information. Appendix A.1 discusses the distribution of a key in a communicating group (i.e., the group members already share a KWK and DAK). Appendix A.2 discusses the distribution of keys using a KDC; Appendix A.3 discusses the establishment of a communicating group using a KDC; and Appendix A.5 discusses the establishment of a communicating group using a multiple-center group to generate and distribute keys to the communicating group.

[42] This could, for example, be a list of entity IDs or a number assigned to a communicating group.

5.3.2 Key Transfers

Keying material is transferred from one entity (i.e., the sender) to one or more other entities (i.e., the receiver(s)). The keying material is sent as *key transfer* information from a KDC, KTC or agent of a multiple-center group to one of its subscribers, or from one member of a communicating group to one or more other members of that group.

Keying material transported in a message containing *key transfer* information **shall** be wrapped using a KWK shared between the sender and the intended receiver. Keying material to be transported in the *key transfer* information **shall** be cryptographically protected as follows:

Let $KWK_{S,R}$ be a KWK shared between the *key transfer* information sender (*S*) and the *key transfer* information receiver (*R*).

If one or more KWKs are included in the *key transfer* information:

- At least one KWK **shall** be wrapped using $KWK_{S,R}$.
- Other KWKs **shall** be wrapped using either $KWK_{S,R}$ or another KWK included in the *key transfer* information, with an indication of the specific KWK that was used.

If one or more KDKs or DKs (i.e., DEKs, DAKs, or AEKs) are included in the *key transfer* information:

- If no KWKs are included in the *key transfer* information, then the KDK(s) and/or DKs **shall** be wrapped using $KWK_{S,R}$
- If KWKs are included in the *key transfer* information, the KDK(s) and/or DK(s) s**hall** be wrapped using either $KWK_{S,R}$ or a KWK included in the *key transfer* information, with an indication of the specific KWK that was used.

Key transfer information **shall** include the wrapped keying material (*wrapped_keys*), an indication of the communicating group members, and provide integrity protection for the entire message and authentication of the entity sending the keying material using an authentication key (*auth_key*) that is included with the wrapped keys and used to generate an authentication code (*auth_code*) that is generated on the message (see Section 5.4):

key_transfer(*wrapped_keys*; *communicating_group*; *auth_code*).

Appendix A provides multiple examples of using *key transfer* information in various scenarios.

5.3.3 Translation Requests

A *translation request* transfers keying material from one entity to a second entity for translation of that keying material for delivery to a third entity. Use cases include the following:

a) The first entity is a KTC subscriber, who generates keys and sends the *translation request* to a KTC (the second entity), who is being requested to translate the keying material for another KTC subscriber (the third entity). An example is provided in Appendix A.4.

b) The first entity is a subscriber of a center that serves as an agent to a multiple-center group (the second entity). The group is being requested to translate the keying material for a third entity, who is presumed to be a subscriber of some center within the multiple-center group. An example is provided in Appendix A.6.

c) The sender of the *translation request* is a center within a multiple-center group, and the receiver is another center within the group. The request may need to be forwarded until a center is found that can perform the requested translation for the ultimate recipient of the keying material (the third party, a subscriber of some center within the multiple-center group). Examples are provided in Appendices A.5 and A.6.

In order to send *translation request* information, the sending entity **shall** share a KWK with the receiver of the

translation request (e.g., a KTC or agent center within a multiple-center group). In order to fulfill the request, a KTC or another center within the multiple-center group **shall** share a KWK with the intended ultimate recipient of the keying material.

A message containing *translation request* information **shall** include wrapped keying material (*wrapped_keys*) and indicate who requested the translation (*requester*) and the communicating group that will use the keying material. Integrity protection **shall** be provided using an authentication key (*auth_key*) generated for each message containing *translation request* information for computing an authentication code (*auth_code*) on the message (see Section 5.4).

translation_request(*wrapped_keys*; *requester*; *communicating group*; *auth_code*).

5.3.4 Revocation Request

Revocation request information is used to request the destruction of the operational and backup copies of keying material. Keys may be revoked by any entity authorized to do so (e.g., authorized in an organization's security policy; by agreement among communicating group; or in an agreement with a KDC, KTC or multiple-center group). The *revocation request* and corresponding *revocation confirmation* information (see Section 5.3.5) may be forwarded if required.

The *revocation request* information **shall** identify the keying material to be destroyed (*key_list*) and provide for the authentication and authorization of the entity requesting the destruction (*requester*) and the integrity of the message using an authentication code (*auth_code*) that is generated using an authentication key (*auth_key*). The authentication key **shall** be newly generated for the message containing the *revocation request* information if the sender can generate keys, and the sender (S) and receiver (R) share a KWK ($KWK_{S,R}$). Otherwise, the authentication key **shall** be a key already shared by the sender and receiver.

In either case, the authentication key (*auth_key*) **shall** be used to compute an authentication code (*auth_code*) on the message (see Section 5.4):

(1) If a newly generated authentication key is used, then:

revocation_request(*key_list*; *requester*; *wrapped_auth_key*; *auth_code*),

where *wrapped_auth_key* = WRAP($KWK_{S,R}$, *auth_key*).

(2) If the sender cannot generate keys, or a KWK is not shared between the sender and receiver:

revocation_request(*key_list*; *requester*; *auth_key_ID*; *auth_code*),

where *auth_key_ID* is used to identify the key used to compute the authentication code (*auth_code*).

Any keys shared between the sender and the receiver(s) may be revoked. If a key at a given layer is revoked, all keys below it in the key hierarchy **shall** be revoked. For example, if a keying relationship was established by a KWK, and the KWK was used later to wrap a KDK, then the revocation of that KDK also revokes all data keys and other KDKs derived from that KDK.

Note that if all Layer 1 keys shared with the receiver(s) are revoked, the relationship among the sender and those receivers is terminated.

Archived copies of a revoked key may be retained if stored in a secure archive facility.

Examples of using revocation requests are provided in Appendix A.8.

5.3.5 Revocation Confirmation

A message containing *revocation confirmation* information provides notification that keys were destroyed as requested in previously received *revocation request* information. The *revocation confirmation* information **shall** indicate the message containing the *revocation request* information to which it is responding (*revocation_request_id*) and provide for the authentication of the entity sending the confirmation and a method of detecting the integrity of the message containing the *revocation confirmation* information using an authentication code (*auth_code*) computed using the authentication key used for the message containing the *revocation request* information. An indication of the key(s) that were destroyed (*list_of_revoked_keys*) **shall** also be included if this can be accomplished without introducing security weaknesses:

revocation_confirmation(*revocation_request_id*; *list_of_revoked_keys*; *auth_code*).

Examples of using revocation confirmations in response to revocation requests are provided in Appendix A.8.

5.3.6 Acknowledgements

An acknowledgement is used to report the receipt of a message without communication errors or other reasons for not acting upon the received message. An acknowledgement is appropriate when:

a) A message containing *key-generation request* information is received, but the request is forwarded to another entity (e.g., a KDC or multiple-center group); in this case, the recipient of the *key-generation request* information cannot generate the requested keying material.

b) A message containing *key tran*sfer information has been received correctly;

c) A message containing *translation request* information is received, but the request is forwarded to another entity.

A message containing *acknowledgement* information **shall** indicate the communication being acknowledged (*previous_message_id*) and provide for the authentication of the entity sending the message and a method of detecting its integrity using a previously established authentication key (*auth_key*) to generate an authentication code (*auth_code*) (see Section 5.4):

acknowledgement(*previous_msg_id*; *auth_code*).

1. If the *acknowledgement* information is sent in response to a message containing *key-generation request* information, the sender (of the *acknowledgement* information)

presumably does not have a key-generation capability (otherwise, a newly generated key would be returned). In this case, an authentication key **shall** use a previously established key for the purpose.

2. If the *acknowledgement* is sent in response to messages containing *key transfer* or *translation request* information, the authentication code **shall** be generated using the authentication key (*auth_key*) used for the message being acknowledged.

Multiple examples of the use of messages containing *acknowledgement* information are provided in Appendix A.

5.3.7 Error Reports

An error message is used to report an error in the previously received key-establishment message. The error message is used to notify the sender of the previous message that the receiver could not act on the previous message information because of an error (e.g., the authentication code for the received message could not be verified, or the request cannot be fulfilled).

The *error report* information in the message **shall** indicate the previous message that is in error, provide for the authentication of the entity sending the error message and a method of detecting its integrity using a previously established authentication key (*auth_key*) to compute an authentication code (*auth_code*) on the message (see Section 5.4). An indication of the specific error (*error_type*) **shall** also be provided if this can be accomplished without introducing weaknesses in the protocol:

error_report(*previous_message_id*; *error_type*; *auth_code*).

Multiple examples of the use of messages containing *error report* information are provided in Appendix A.

5.4 Authentication Codes in Key-Establishment Messages

As required in Section 5.1, an authentication code is required on all key-establishment messages. The authentication code is generated on the entire message (with the exception of the authentication code itself) using an **approved** authentication algorithm and the authentication key (*auth_key*). The authentication algorithm may be indicated in the key-establishment message, negotiated between the sender and receiver or determined by the communications protocol.

The authentication key may have been previously established between the sender and receiver (e.g., manually or in a previous message) or may be carried in the message itself (e.g., in a message containing *key transfer* or *translation request* information). If carried in a message, the sender (of the outgoing message) **shall** wrap the authentication key using a KWK either contained in the message or already shared between the sender and receiver; a receiver must unwrap the key in order to verify the authentication code in the message.

The authentication code **shall** be verified by a receiver before taking action on the key-establishment information in any received message (see Section 5.1).

- If the verification fails, a message containing *error report* information **shall** be sent to the message sender (see Section 5.3.7).

- If the verification is successful and another message is not to be immediately sent to the message sender in response to a request, an acknowledgement **shall** be sent to the message sender (see Section 5.3.6).

5.5 Revocation and Destruction

General guidance regarding the destruction of cryptographic keys is provided in SP 800-57 Part 1 and in SP 800-88. Except for archival purposes, when keys have been compromised, suspected of having been compromised, or revoked, they **shall** be physically or logically destroyed so that they cannot be recovered (e.g., by overwriting with another key or a constant value).

For a keying relationship (e.g., between members of a communicating group or between a center and a subscriber), if a key is revoked, all keys that are lower in that key's hierarchy **shall** be revoked. For example, if a Layer 1 KWK shared by a communicating group is used to protect a KDK distributed within the group, a revocation of the KDK requires the destruction of that KDK and all data keys and other KDKs derived from that KDK; a revocation of the Layer 1 KWK requires the destruction of the KWK and all keys below it in the key hierarchy. If a Layer 1 key is revoked, and there is no other Layer 1 key to continue a keying relationship (e.g., for a communicating group), then the relationship **shall** be terminated.

Cryptographic keys **shall** be destroyed in accordance with SP 800-88. FIPS140 contains suggested methods for the destruction of keying materials within cryptographic modules.

- The destruction of keys **shall** be accomplished under conditions of full accountability, with appropriate records retained for audit trail purposes. Note that some keys (e.g., derived keys, and some other locally generated one-time or short-term keys) are not usually recorded and may be exempt from accounting rules. See SP 800-57, Part 2, for accounting guidelines for cryptographic keys.

Appendix A: Example Scenarios

This appendix contains examples using the key-establishment information specified in Section 5 in various scenarios.

Note that error handling of received acknowledgements is often suggested, but is not included in detail.

A.1 Communicating Group Key Transfer

In this example, a communicating group was established between two entities as discussed in Section 4.2. As shown in Figure A.1, Entities A and B share a KWK to be used for key wrapping, and a DAK to be used for authentication when needed (i.e., $KWK_{A,B}$ and $DAK_{A,B}$); these two keys are the Layer 1 keys shared by the group. In this example, Entity A can generate keys, but Entity B cannot.

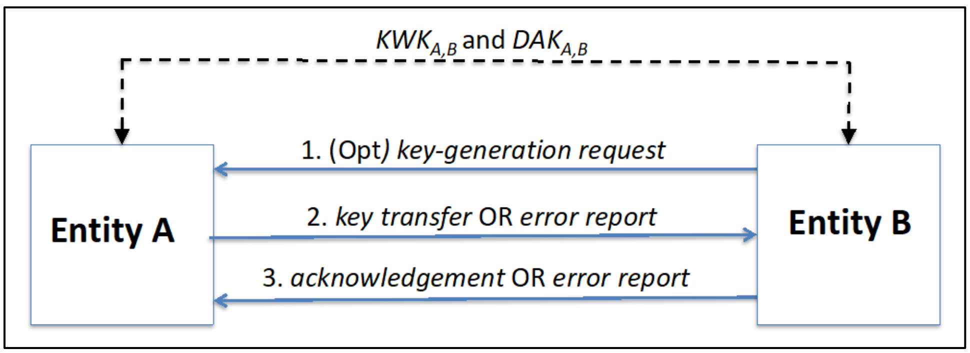

Figure A.1: Key-Generation Request and Key Transfer in a Communicating Group

1. If Entity B would like to exchange information with Entity A, then Entity B could, for example, send a *key-generation request* to Entity A asking for an AEK to be used with the AES-128 block cipher:

   key-generation_request(AEK: AES-128; *communicating group*: Entity_A, Entity_B; $auth\_code_1$).

   where $auth\_code_1$ is generated using the shared DAK ($DAK_{A,B}$).

2. Entity A generates keys in response to requests from Entity B or of its own volition.

   (a) If Entity A receives a *key-generation request*: Entity A attempts to verify that the message containing the *key-generation request* (see step 1) was correctly received from a member of the communicating group (i.e., Entity B). If the verification fails, then an error message is sent to Entity B containing *error report* information, and further interaction is terminated.

   error_report(*previous_message_id*; *error_type*; $auth\_code_2$),

   where *previous_message_id* is the ID for the *key-generation_request* (see step 1), the *error_type* is the type of error, and $auth\_code_2$ is generated using $DAK_{A,B}$.

   Entity B could choose to resend the *key-generation request* (see step 1).

(b) When Entity A generates a key for Entity B (either in response to a verified request from Entity B or of its own volition), a transaction authentication key is also generated (*Transaction_DAK*$_{A,B}$) rather than using the already-shared authentication key (i.e., *DAK*$_{A,B}$). Entity A wraps the key to be sent (K) and the transaction authentication key using the shared KWK (*KWK*$_{A,B}$):

$$wrapped\_keys = \mathrm{WRAP}(KWK_{A,B},\ K \parallel Transaction\_DAK_{A,B}).$$

(c) The *wrapped_key* is sent to Entity B in a message containing *key transfer* information:

key_transfer(*wrapped_keys*; *communicating_group*: Entity_A, Entity_B; *auth_code*$_3$),

where *auth_code*$_3$ is generated using *Transaction_DAK*$_{A,B}$.

3. When the *key transfer* information is received, Entity B sends either an *acknowledgement* or an *error report*.

(a) Entity B unwraps the wrapped keys and uses *Transaction_DAK*$_{A,B}$ to attempt a verification of the message. If the verification fails, an error message is sent to Entity A containing *error report* information:

error_report(*previous_message_id*; *error_type*; *auth_code*$_4$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 2c), the *error_type* is the type of error, and *auth_code*$_4$ is generated using *DAK*$_{A,\,B}$. Since the message had an error, *Transaction_DAK*$_{A,\,B}$ may not have been received correctly, so *DAK*$_{A,\,B}$ is used as the authentication key.

Entity A may resend the *key transfer* information (see step 2c).

(b) If the verification of the authentication code is successful, a message containing *acknowledgment* information is sent:

acknowledgement(*previous_msg_id*; *auth_code*$_5$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 2c), and *auth_code*$_5$ is generated using *Transaction_DAK*$_{A,\,B}$.

Note that for the sake of brevity, Entity A's receipt and handling of the *acknowledgement* information is not discussed in detail here. However, if the message containing the *acknowledgement* information cannot be verified, then Entity A could send an error message to Entity B, and Entity B could resend the *acknowledgement* information (see step 3b).

A.2 Using a KDC to Distribute Keys to an Already-Established Communicating Group

Entities A and B are members of a communicating group; they share *KWK*$_{A,\,B}$ and *DAK*$_{A,\,B}$ as their Layer 1 keys. Neither entity can generate keying material. However, they are subscribers of the same KDC.

Entity A shares *KWK*$_{A,\,KDC}$ and *DAK*$_{A,\,KDC}$ with the KDC; Entity B shares *KWK*$_{B,\,KDC}$ and *DAK*$_{B,\,KDC}$ with the KDC. Additional keying material can be generated by the KDC and distributed to A and B. Figure A.2a and Figure A.2b depict two alternatives, differing only in how the keys are distributed to A and B after generation by the KDC.

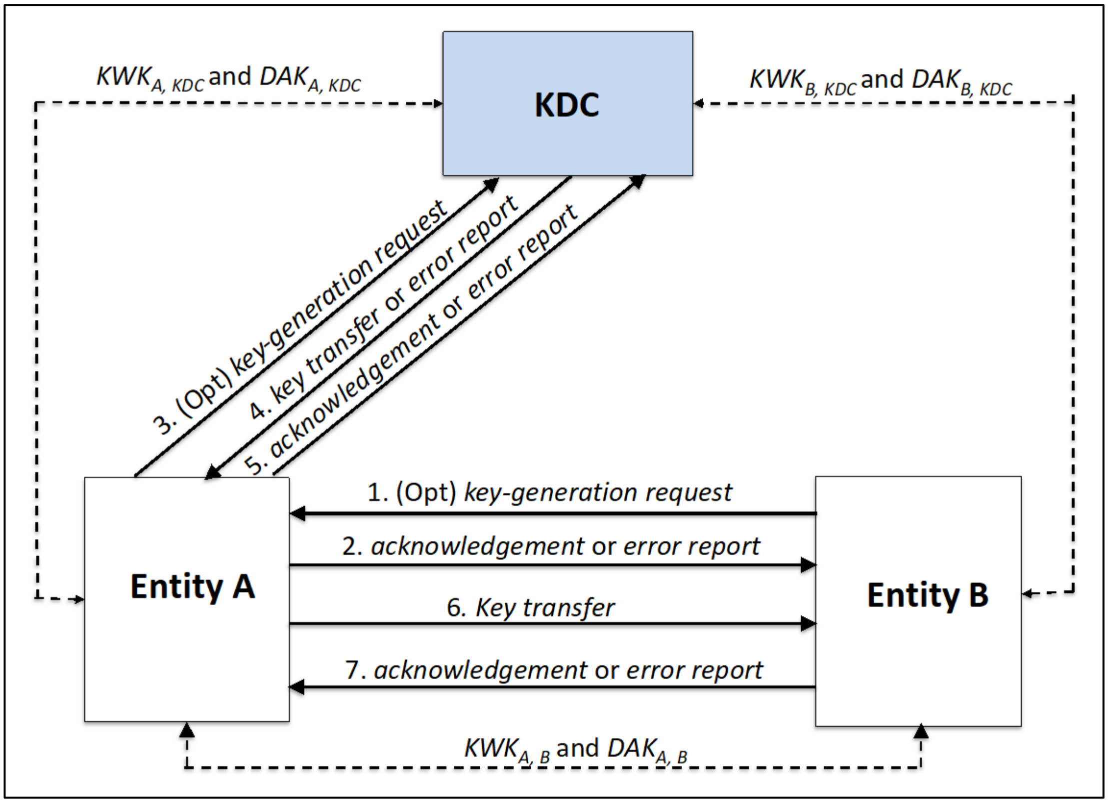

Figure A.2a: Using a KDC (Alternative 1)

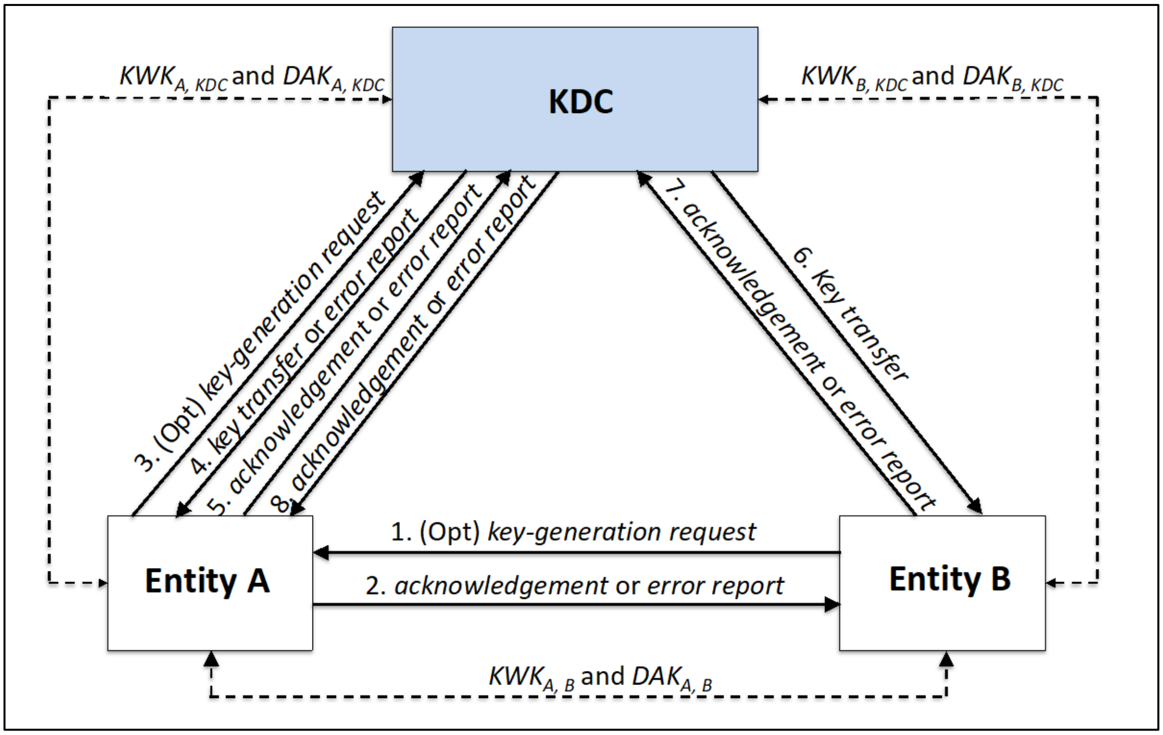

Figure A.2b: Using a KDC (Alternative 2)

1. Entity B may optionally send a *key-generation request* to Entity A asking for a KWK to be used with the AES-128 block cipher.

   key-generation_request(KWK: AES-128; *communicating_group*: Entity_A, Entity_B; $auth\_code_1$).

   where $auth\_code_1$ is generated using the shared DAK ($DAK_{A,B}$).

2. (a) If Entity A receives a *key-generation request* (see step 1): Entity A attempts to verify that the message containing the *key-generation request* was correctly received from a member of the communicating group (i.e., Entity B). If the verification fails, then an error message is sent to Entity B containing *error report* information, and further interaction is terminated.

   error_report(*previous_message_id*; *error_type*; $auth\_code_2$),

   where *previous_message_id* is the ID for the message containing the *key-generation_request* (see step 1), the *error_type* is the type of error, and $auth\_code_2$ is generated using $DAK_{A, B}$.

   Entity B could choose to resend the *key-generation request* (see step 1).

   (b) If the verification of the authentication code is successful, a message containing *acknowledgment* information is sent to Entity B:

   acknowledgement(*previous_msg_id;* $auth\_code_3$),

   where *previous_message_id* is the ID for the message containing the *key-generation request* (see step 1) and $auth\_code_3$ is generated using $DAK_{A, B}$.

3. Entity A sends a *key-generation request* to the KDC asking for a KWK to be used with the AES-128 block cipher and shared with Entity B:

   key-generation_request(KWK: AES-128; *communicating_group*: Entity_A, Entity_B; $auth\_code_4$),

   where $auth\_code_4$ is generated on the message containing the *key-generation request* using the DAK shared with the KDC ($DAK_{A,\ KDC}$).

4. (a) The KDC attempts to verify $auth\_code_4$ using the DAK shared with Entity A ($DAK_{A,\ KDC}$). If the verification fails or if Entity B is not a subscriber of the KDC, *error report* information is returned to Entity A in an error message, and the process is terminated.

   error_report(*previous_message_id*; *error_type*; $auth\_code_5$),

   where *previous_message_id* is the ID for the message containing the *key-generation request* (see step 3), *error_type* is the type of error, and $auth\_code_5$ is generated using $DAK_{A,\ KDC}$.

   If the error was because of a verification failure, Entity A may choose to resend the *key-generation request* (see step 3). If a *key-generation request* was received from Entity B (see step 1), Entity A may notify Entity B of the problem in an error message (not shown in the figures).

   (b) If the verification of the *key-generation_request* is successful, and Entity B is a subscriber of the KDC, the KDC generates the requested key (K) and an authentication key (*Transaction_auth_key*), and wraps one copy of the keys using the KWK shared with Entity A ($KWK_{A,\ KDC}$) and another copy of the keys using the KWK shared with Entity B ($KWK_{B,\ KDC}$).

   $$Entity\_A\_wrapped\_keys = \text{WRAP}(KWK_{A,\ KDC}, K \,\|\, Transaction\_auth\_key)$$

   $$Entity\_B\_wrapped\_keys = \text{WRAP}(KWK_{B,\ KDC}, K \,\|\, Transaction\_auth\_key).$$

At this point in the process, go to Alternative 1 or Alternative 2.

Alternative 1 (using Figure A.2a): Continuing at step 4 (c).

   (c) The two copies of the wrapped keys are sent to Entity A in a *key transfer* message:

   key_transfer(*Entity_A_wrapped_keys*, *Entity_B_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; $auth\_code_6$),

   where $auth\_code_6$ is generated using *Transaction_auth_key*.

5. (a) Upon receiving the *key_transfer* information, Entity A extracts its copy of the wrapped keys from the message (see step 4c), unwraps the keys using the KWK shared with the KDC ($KWK_{A,\ KDC}$), and checks the message's authentication code ($auth\_code_6$) using the unwrapped transaction authentication key (*Transaction_auth_key*).

   (b) If the verification of the authentication code fails, then an error-report message is sent to the KDC containing *error_report* information:

   error_report(*previous_message_id*; *error_type*; $auth\_code_7$), where *previous_message_id* is the ID for the message containing the *key_transfer* information (see step 4c), the *error_type* is the type of error, and $auth\_code_7$ is generated using $DAK_{A,\ KDC}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the message may not be correct, so $DAK_{A,\ KDC}$ is used as the authentication key.

The KDC may choose to resend the *key_transfer* information (see step 4c).

(c) If the verification of the authentication code is successful, a message containing *acknowledgment* information is sent to the KDC:

acknowledgement(*previous_msg_id*; $auth\_code_8$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 4c), and $auth\_code_8$ is generated using *Transaction_auth_key*.

6. Entity A creates and sends a message to Entity B containing *key transfer* information that includes Entity B's copy of the wrapped keys:

key_transfer(*Entity_B_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; $auth\_code_9$),

where $auth\_code_9$ is computed on the message using the transaction authentication key received from the KDC (*Transaction_auth_key*).

7. Entity B sends either *acknowledgement* or *error_report* information to Entity A after receiving the message.

(a) Entity B extracts the wrapped keys from the received *key transfer* information (see step 6), unwraps the keys using the KWK shared with the KDC ($KWK_{B,\ KDC}$), and checks the message's authentication code ($auth\_code_9$) using the unwrapped authentication key (*Transaction_auth_key*).

(b) If the verification of the authentication code fails, then an error message is sent to Entity A containing *error report* information.

error_report(*previous_message_id; error_type*; $auth\_code_{10}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 6), the *error_type* is the type of error, and $auth\_code_{10}$ is computed on the message using $DAK_{A,\ B}$. Since the received message had an error, *Transaction_auth_key* may have been received incorrectly, so $DAK_{A,\ B}$ is used as the authentication key.

Entity A may resend the *key transfer* message (see step 6).

(c) If the verification of the authentication code is successful, then a message containing *acknowledgement* information is sent to Entity A:

acknowledgement(*previous_msg_id*; $auth\_code_{11}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 6), and $auth\_code_{11}$ is generated using *Transaction_auth_key.*

Note that for the sake of brevity, Entity B's receipt and handling of the *acknowledgement* information is not discussed in detail here. However, if the message containing the *acknowledgement* information cannot be verified, then Entity B could send an error message to Entity A, and Entity A could resend the *acknowledgement* information (see step 7c).

At this point, both Entity A and Entity B know that they have successfully received the new KWK.

Alternative 2 (using Figure A.2b): Continuing at step 4 (c).

(c) The KDC sends a message containing *key transfer* information to Entity A with the appropriate copy of the wrapped keys:

key_transfer(*Entity_A_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; $auth\_code_{12}$),

where $auth\_code_{12}$ is generated using *Transaction_auth_key*.

5. (a) Entity A extracts the wrapped keys from the received message (see step 4c), unwraps the keys using the KWK shared with the KDC ($KWK_{A,\ KDC}$), and checks the message's authentication code ($auth\_code_{12}$) using the unwrapped authentication key (*Transaction_auth_key*).

(b) If the verification of the authentication code fails, then an error message containing *error report* information is sent to the KDC.

error_report(*previous_message_id*; *error_type*; $auth\_code_{14}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 4c), the *error_type* is the type of error, and $auth\_code_{14}$ is generated using $DAK_{A,\ KDC}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the message may not be correct, so $DAK_{A,\ KDC}$ is used as the authentication key.

The KDC may resend the message containing the *key transfer* information (see step 4c).

(c) If the verification of the authentication code is successful, then a message containing *acknowledgement* information is sent to the KDC.

acknowledgement(*previous_msg_id*; $auth\_code_{15}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 4c), and $auth\_code_{15}$ is generated using *Transaction_auth_key*.

6. If the KDC receives an acknowledgement from Entity A (indicating that Entity A has the keying material), the KDC sends a message containing *key transfer* information to Entity B with the appropriate copy of the wrapped keys:

key_transfer(*Entity_B_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; $auth\_code_{13}$),

where $auth\_code_{13}$ is generated using *Transaction_auth_key*.

7. (a) Entity B extracts the wrapped keys from the *key transfer* information in the received message (see step 4c), unwraps the keys using the KWK shared with the KDC ($KWK_{B,\ KDC}$), and checks the message's authentication code ($auth\_code_{13}$) using the unwrapped authentication key (*Transaction_auth_key*).

(b) If the verification of the authentication code fails, then an error message is sent to the KDC containing *error report* information.

error_report(*previous_message_id*; *error_type*; $auth\_code_{16}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 4c), the *error_type* is the type of error, and $auth\_code_{16}$ is generated using $DAK_{B,\,KDC}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *key transfer* information may not be correct, so $DAK_{B,\,KDC}$ is used as the authentication key.

The KDC may resend the message (see step 4c).

(c) If the verification of the authentication code is successful, then a message containing *acknowledgement* information is sent to the KDC.

acknowledgement(*previous_msg_id*; $auth\_code_{17}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 4c), and $auth\_code_{17}$ is generated using *Transaction_auth_key*.

Note that for the sake of brevity, the KDC's receipt and handling of the *acknowledgement* information is not discussed in detail here. However, if the message containing the *acknowledgement* information cannot be verified, then the KDC could send an error message to Entity B, and Entity B could resend the *acknowledgement* information (see step 7c).

8. At this point, the KDC and Entity B know that Entities A and B share the new keys (because of the protocol flow; see steps 5 and 6), but Entity A has has not been notified of this fact.

The KDC sends an a message to Entity A containing acknowledgement information indicating that Entity B has successfully received the new keys:

acknowledgement(*previous_msg_id*; $auth\_code_{18}$),

where *previous_message_id* is the ID for the message containing the *key generation request* (see step 3), and $auth\_code_{18}$ is generated using *Transaction_auth_key*.

Note that for the sake of brevity, Entity A's receipt and handling of the *acknowledgement* information is not discussed in detail here. However, if the message containing the *acknowledgement* information cannot be verified, then Entity A could send an error message to the KDC, and the KDC could resend the *acknowledgement* information (see step 7c).

If the acknowledgement message is successfully verified, Entities A and B now know that they share the new keys.

A.3 Using a KDC to Establish a Communicating Group

Entities A and B do not share keys, but a decision has been made that they need to communicate securely. This can be done using the services of a KDC to form a communicating group.

Both entities share keys with the same KDC. Entity A shares $KWK_{A,\,KDC}$ and $DAK_{A,\,KDC}$ with the KDC; Entity B shares $KWK_{B,\,KDC}$ and $DAK_{B,\,KDC}$ with the KDC (see Figure A.3).

In this example, Entity A (the requesting subscriber) requests the KDC to generate keys to be shared with Entity B in order to form a communicating group. The KDC generates the requested keys and sends them to Entity B. After receiving an acknowledgement from Entity B that the keys have been received correctly, the KDC sends the keys to Entity A. When an acknowledgement of correct receipt has been received from Entity A, the communicating group is considered to be established.

Variants of this scenario are possible but would require other message flows. For example, when the KDC begins the process by sending keys to a subset of subscribers without receiving a request, one or more additional messages may be required to provide assurance to the entities that they do indeed share keys and form a communicating group (e.g., messages from the KDC or between the entities). If communicating groups are larger than two entities, then additional messages will be required to distribute the keys to the additional entities and to provide mutual assurance that the group has been completely established.

The steps following Figure A.3 discuss the case where one subscriber (Entity A) requests the KDC to generate keys for Entity B.

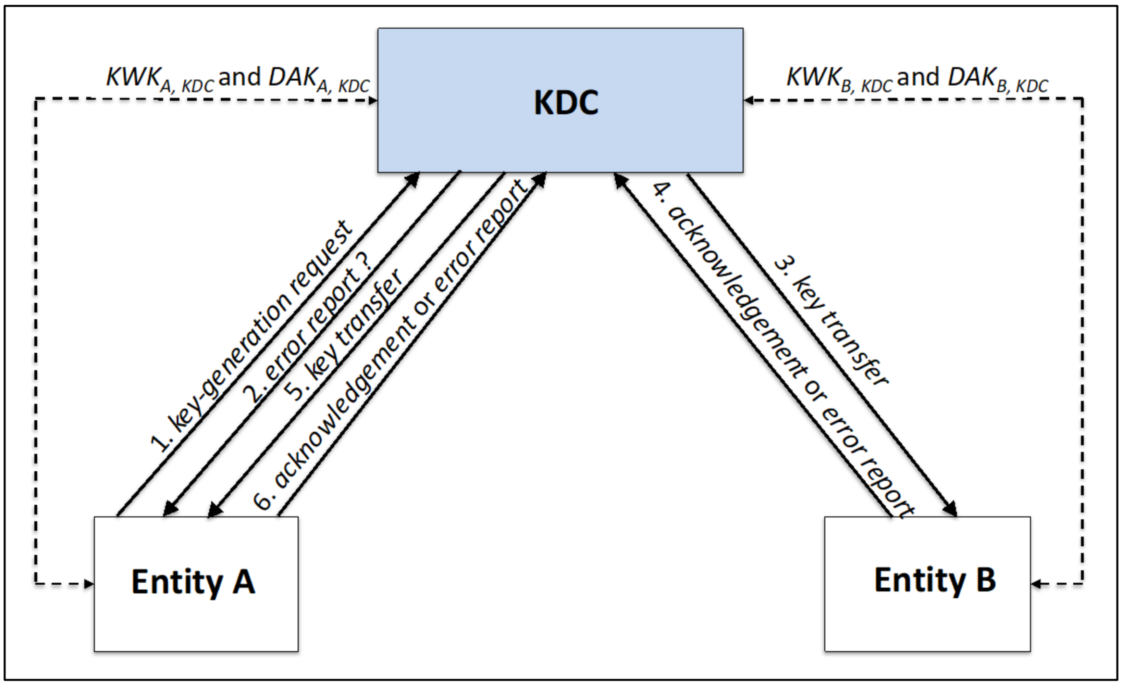

Figure A.3: Establishing a Communicating Group Using a KDC

1. Entity A sends a *key-generation request* to the KDC asking for a KWK and DAK to share with Entity B:

   key-generation_request(KWK, DAK; *communicating_group*: Entity_A, Entity_B; *auth_code*$_1$),

   where *auth_code*$_1$ is generated on the message containing the *key-generation request* using the DAK shared with the KDC ($DAK_{A,\ KDC}$).

2. The KDC attempts to verify that the message containing the *key-generation request* was correctly received and that Entity B is a KDC subscriber. If the verification fails, then an error message is sent to Entity A containing *error report* information, and further interaction is terminated.

   error_report(*previous_message_id*; *error_type*; $auth\_code_2$),

   where *previous_message_id* is the ID for the message containing the *key-generation_request* (see step 1), the *error_type* is the type of error, and $auth\_code_2$ is generated using $DAK_{A,\ KDC}$.

   Entity A could choose to resend the *key-generation request* (see step 1).

3. (a) The KDC generates the requested keying material to be shared between Entities A and B ($KWK_{A,\ B}$ and $DAK_{A,\ B}$) and a transaction authentication key (*Transaction_auth_key*). In this example, $KWK_{A,\ B}$ is intended to be a Layer 1 key, with $DAK_{A,\ B}$ beneath it in the key hierarchy. This will allow a termination of the communicating group in the future by revoking just the Layer 1 key (see Section 5.5 and Appendix A.8, Example 1). Therefore, $KWK_{A,\ B}$ is wrapped using the KWK shared with the intended receiving entity (A or B), and $DAK_{A,\ B}$ is wrapped using $KWK_{A,\ B}$. For efficiency reasons, the transaction authentication key is wrapped using the KWK shared with the intended receiving entity:

   $$Entity\_A\_wrapped\_KWK\_and\ auth\_key = \text{WRAP}(KWK_{A,\ KDC},\ KWK_{A,\ B} \mid\mid Transaction\_auth\_key);$$

   $$Entity\_B\_wrapped\_KWK\_and\ auth\_key = \text{WRAP}(KWK_{B,\ KDC},\ KWK_{A,\ B} \mid\mid Transaction\_auth\_key);$$

   $$wrapped\_DAK = \text{WRAP}(KWK_{A,\ B},\ DAK_{A,\ B}).$$

   (b) The KDC creates and sends a message containing *key transfer* information to Entity B (see step 3a):

   key_transfer(*Entity_B_wrapped_KWK_and auth_key, wrapped_DAK*; *communicating_group*: Entity_A, Entity_B; $auth\_code_3$),

   where $auth\_code_3$ are generated using *Transaction_auth_key*.

4. (a) Entity B extracts and unwraps $KWK_{A,\ B}$ and *Transaction_auth_key* using the KWK shared with the KDC ($KWK_{B,\ KDC}$), and checks the message's authentication code ($auth\_code_3$) using the unwrapped authentication key (*Transaction_auth_key*).

   (b) If the verification of the authentication code fails, or if Entity B does not wish to establish a communicating group with Entity A, then an error message is sent to the KDC containing *error report* information.

   error_report(*previous_message_id*; *error_type*; $auth\_code_4$),

   where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 3b), the *error_type* is the type of error, and $auth\_code_4$ is generated using $DAK_{B,\ KDC}$. Note that when there is an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *key transfer* information may not be correct, so $DAK_{B,\ KDC}$ is used as the authentication key. Also, if Entity B does not want to establish a communicating group with Entity A, using the authentication key received in the *key transfer* information (*Transaction_auth_key*) may not be desirable.

The KDC may resend the *key transfer* message (see step 3b). Alternatively, the KDC may send *error report* information to Entity A indicating that the keys could not be established with Entity B (not shown in the figure).

(c) If the verification of the authentication code is successful, and Entity B wants to establish a communicating group with Entity A, then Entity B unwraps $DAK_{A,\,B}$ using $KWK_{A,\,B}$.

(d) Entity B sends a message to the KDC containing *acknowledgement* information:

acknowledgement(*previous_msg_id*; *auth_code*$_5$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 3c), and *auth_code*$_5$ is generated using *Transaction_auth_key*.

5. Upon receiving a message from Entity B containing the *acknowledgement* information and verifying its correct receipt (left as an exercise for the reader!), the KDC prepares and sends key transfer information to Entity A:

key_transfer(*Entity_A_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; *auth_code*$_6$),

where *auth_code*$_6$ are generated using *Transaction_auth_key*.

6. (a) Entity A extracts the wrapped keys from the received message (see step 5), unwraps the keys using the KWK shared with the KDC ($KWK_{A,\,KDC}$), and checks the message's authentication code (*auth_code*$_6$) using the unwrapped authentication key (*Transaction_auth_key*).

(b) If the verification of the authentication code fails, then an error message containing *error report* information is sent to the KDC.

error_report(*previous_message_id*; *error_type*; *auth_code*$_7$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 5), the *error_type* is the type of error, and *auth_code*$_7$ is generated using $DAK_{A,\,KDC}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the message may not be correct, so $DAK_{A,\,KDC}$ is used as the authentication key.

The KDC may resend the message containing the *key transfer* information (see step 5).

(c) If the verification of the authentication code is successful, then a message containing *acknowledgement* information is sent to the KDC.

acknowledgement(*previous_msg_id*; *auth_code*$_8$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 5), and *auth_code*$_8$ is generated using *Transaction_auth_key*.

At this point, Entities A and B share a KWK and DAK as members of the same communicating group. However, only Entity A knows for sure that the keys are shared (because of the order that the keys were distributed by the KDC in this example). Entity B could be notified of this fact in a couple of ways:

- By Entity A or B sending a cryptographically protected message to the other party (e.g., protected using the newly established KWK and DAK); or
- By the KDC sending acknowledgement information to Entity B indicating that the establishment of the communicating group has been completed.

A.4 Using a KTC to Establish a Communicating Group

Entities A and B do not share keys, but a decision has been made that they need to communicate securely. This can be done using the services of a KTC to form a communicating group.

In this example, Entity A generates Layer 1 keys to be shared with Entity B and sends them to the KTC for translation. The KTC translates the keys and sends them to Entity B. After receiving an acknowledgement that the keys have been received correctly by Entity B, the KTC sends an acknowledgement of correct receipt to Entity A; the communicating group is then considered to be established.

Variants of this scenario are possible but would require other message flows. For example, if communicating groups are larger than two entities, then additional messages will be required to distribute the keys to the additional entities and to provide mutual assurance that the group has been completely established.

Figure A.4 shows the use of a KTC by two entities (A and B) that want to establish a communicating group. Both entities are subscribers of the same KTC; Entity A shares $KWK_{A, KTC}$ and $DAK_{A, KTC}$ with the KTC; Entity B shares $KWK_{B, KTC}$ and $DAK_{B, KTC}$ with the KTC. In this case, Entity A can generate keying material. For this example, the KTC does not generate keys.

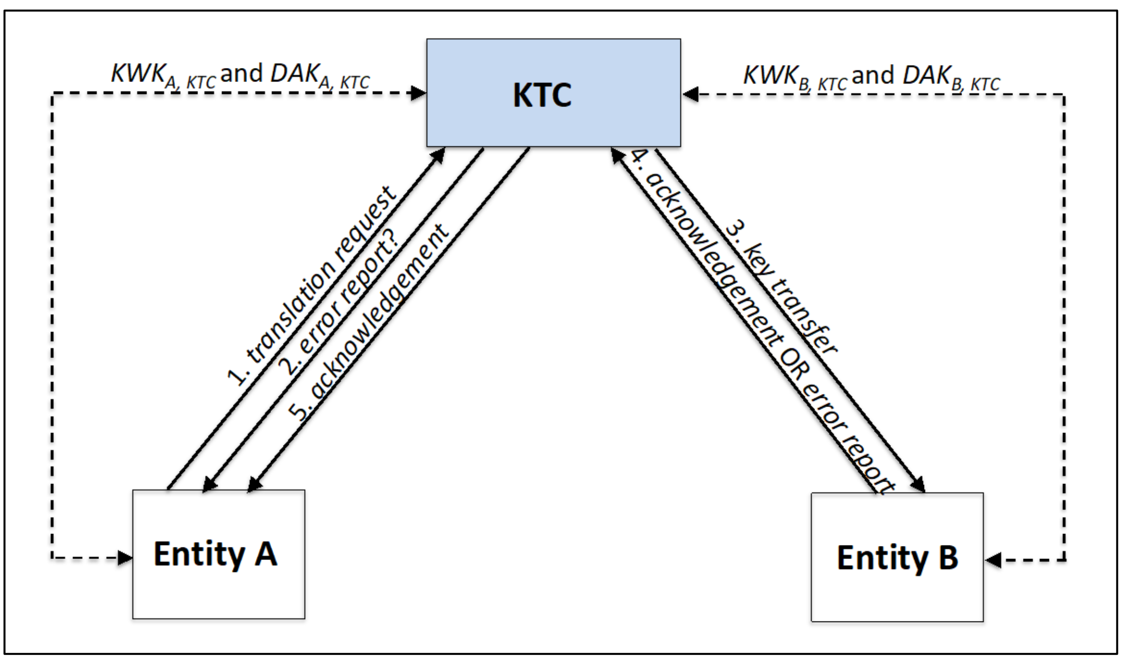

Figure A.4: Using a KTC

1. (a) Entity A generates a KWK ($KWK_{A, B}$) and an authentication key ($DAK_{A, B}$) to be sent to Entity B, and another authentication key (*Transaction_auth_key*) to provide authentication for the *translation request* to be sent to the KTC.

   (b) Entity A wraps the generated keys using the KWK shared with the KTC ($KWK_{A, KTC}$):

   $$wrapped\_keys = \text{WRAP}(KWK_{A, KTC}, KWK_{A, B} \| DAK_{A, B} \| Transaction\_auth\_key).$$

   Note that in this example, $KWK_{A, B}$ and $DAK_{A, B}$ are wrapped using the same key (i.e., $KWK_{A, KTC}$). In this case, $KWK_{A, B}$ and $DAK_{A, B}$ are both Layer 1 keys.

   (c) Entity A prepares and sends a message containing *translation request* information to the KTC with the wrapped keys:

   translation_request(*wrapped_keys*; *requester*: Entity_A; *sharing_entities*: Entity_A, Entity_B; $auth\_code_1$),

   where $auth\_code_1$ is generated using the message's authentication key (*Transaction_auth_key*).

2. The KTC unwraps the wrapped keying material in the *translation request* information and uses the unwrapped message authentication key (*Transaction_auth_key*) to verify the message. If the verification fails, an error report is sent to Entity A:

   error_report(*previous_message_id*; *error_type*; $auth\_code_2$),

   where *previous_message_id* is the ID for the message containing the *translation request* information (see step 1), the *error_type* is the type of error, and $auth\_code_2$ is generated using $DAK_{A, KTC}$. Note that since there was an error in the message, the wrapped authentication key (*Transaction_auth_key*) in the *translation request* information may not be correct, so $DAK_{A, KTC}$ is used as the authentication key.

   Entity A may choose to resend the *translation request* information (not shown in the figure).

3. (a) If the authentication code is successfully verified, the KTC wraps the received keys for Entity B using the KWK shared with B ($KWK_{B, KTC}$):

   $$wrapped\_keys = \text{WRAP}(KWK_{B, KTC}, KWK_{A, B} \| DAK_{A, B} \| Transaction\_auth\_key).$$

   (b) The KTC prepares and sends a message to Entity B containing the wrapped keys as *key transfer* information and generates $auth\_code_3$ on the message using the received authentication key (*Transaction_auth_key*):

   key_transfer(*wrapped_keys*; *communicating_group*: Entity_A, Entity_B; $auth\_code_3$).

4. (a) Entity B unwraps the received keys and attempts to verify $auth\_code_3$ using the authentication key included in the message containing the *key transfer* information (*Transaction_auth_key*); see step 3b.

   (b) If the verification fails or if Entity B does not want to establish a communicating group with Entity A, a message containing *error report* information is returned to the KTC, and the process is terminated.

   error_report(*previous_message_id*; *error_type;* $auth\_code_4$), where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 3b), *error_type* is the type of error, and $auth\_code_4$ is generated on the error message using $DAK_{B,\ KTC}$. Note that when there is an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *key transfer* information may not be correct, so $DAK_{B,\ KTC}$ is used as the authentication key. Also, if Entity B does not want to establish a communicating group with Entity A, using the authentication key received in the *key transfer* information (*Transaction_auth_key*) may not be desirable.

The KTC may choose to resend the *key transfer* information (not shown in the figure).

(c) If the verification is successful, and Entity B wants to establish a communicating with Entity A, Entity B sends a message containing *acknowledgement* information to the KTC:

acknowledgement(*previous_msg_id*; $auth\_code_5$),

where *previous_message_id* is the ID for the message containing the *key-transfer* information (see step 3b), and $auth\_code_5$ is generated on the message using *Transaction_auth_key*.

5. If the message containing the *acknowledgement* information is received correctly from Entity B[43], then the KTC prepares and sends a message containing *acknowledgement* information to Entity A indicating that the communicating group has been established successfully:

acknowledgement(*previous_msg_id*; $auth\_code_6$),

where *previous_message_id* is the ID for the message containing the *translation request* information (see step 1b), and $auth\_code_6$ is generated using *Transaction_auth_key*.

Note that for the sake of brevity, Entity A's receipt and handling of the *acknowledgement* information is not discussed in detail here. However, if the message containing the *acknowledgement* information cannot be verified, then Entity A could send an error message to the KTC, and the KTC could resend the *acknowledgement* information above.

Note: alternatively, a special-purpose confirmation message could be used to indicate successful communicating-group establishment.

A.5 Using a Multiple-Center Group to Generate a Key for Establishing a Communicating Group

A communicating group may be established using the services of a multiple-center group to generate the Layer 1 keys to be shared by the members of a communicating group. In this example (shown as Figure A.5), the multiple-center group consists of Center 1 and Center 2; these centers share a KWK and a DAK (e.g., $KWK_{1,\ 2}$ and $DAK_{1,\ 2}$). Center 1 is Entity A's agent to the group; they share $KWK_{A,\ 1}$ and $DAK_{A,\ 1}$. Center 2 is Entity B's agent to the group; they share $KWK_{B,\ 2}$ and $DAK_{B,\ 2}$. Entities A and B do not currently share keys.

In this example, Center 1 generates keying material at Entity A's request and sends it to Center 2 for translation for Entity B. Center 2 translates the keying material for Entity B and sends it to B. After receiving an acknowledgement that the keys have been received correctly by Entity B, Center

[43] The handling of errors in the received acknowledgement is left to the reader.

2 sends an acknowledgement of correct receipt to Center 1, who forwards the acknowledgement to Entity A; the communicating group is then considered to be established.

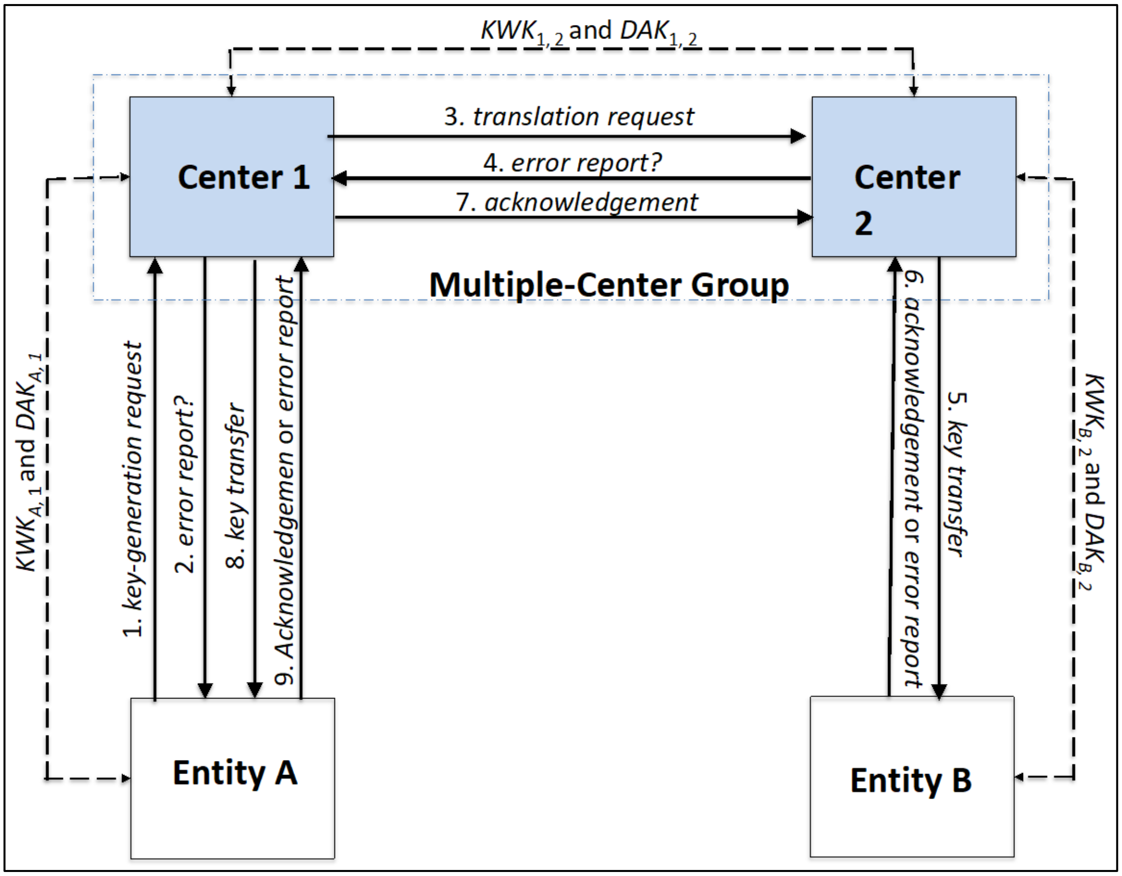

Figure A.5: Establishing a Communicating Group Using a Multiple-Center Group for Key Generation

1. Entity A may optionally send a *key-generation request* to its agent asking for the generation of a KWK and DAK to be used with Entity B.

   key-generation_request(KWK, DAK; *communicating_group*: Entity_A, Entity_B; $auth\_code_1$).

   where $auth\_code_1$ is generated on the message containing the *key_generation request* information using $DAK_{A,1}$.

2. Center 1 attempts to verify $auth\_code_1$ using the DAK shared with Entity A (i.e., $DAK_{A,1}$). If the verification fails, an error message containing *error report* information is returned to Entity A, and the process is terminated.

   error_report(*previous_message_id*; *error_type*; $auth\_code_2$),

   where *previous_message_id* is the ID for the message containing the *key-generation request* information (see step 1), the *error_type* is the type of error, and $auth\_code_2$ is generated using $DAK_{A,1}$.

Entity A may choose to resend the *key-generation request* information (not shown in the figure).

3. (a) If the verification of the message containing the key-*generation request* information is successful, but the other entity identified in the *key-generation request* (Entity B) is not a subscriber of Center 1, then Center 1 suspects that Entity B may be a subscriber of another center in the multiple-center group (i.e., Center 2 in this example).

   (b) Center 1 generates the requested keying material ($KWK_{A,\ B}$ and $DAK_{A,\ B}$) and an authentication key (*Transaction_auth_key*).

   (c) Center 1 needs to send *translation request* information to Center 2, so another authentication key is generated (*Group_Transaction_auth_key*) and wrapped with the keys to be translated using the KWK shared with Center 2 ($KWK_{1,2}$).

   $$Center\_2\_wrapped\_keys = \text{WRAP}(KWK_{1,\,2},\ KWK_{A,\,B}\ ||\ DAK_{A,\,B}\ ||\ Transaction\_auth\_key\ ||\ Group\_Transaction\_auth\_key).$$

   (d) Center 1 prepares and sends a message to Center 2 containing *translation request* information with the wrapped keys:

   translation_request(*Center_2_wrapped_keys*; *requester*: Center_1; *communicating_group*: Entity_A, Entity B; $auth\_code_3$),

   where $auth\_code_3$ is generated using *Group_Transaction_auth_key*.

4. (a) Center 2 unwraps the keys in the *translation request* information using $KWK_{1,2}$ to obtain the authentication key used for the message (*Group_Transaction_auth_key*).

   (b) Center 2 attempts to verify $auth\_code_3$ using the unwrapped authentication key (*Group_Transaction_auth_key*). If the verification fails, or Entity B is not a subscriber of Center 2, a message containing *error report* information is returned to Center 1, and the process is terminated.

   error_report(*previous_message_id*; *error_type*; $auth\_code_4$),

   where *previous_message_id* is the ID for the message containing the *translation request* information (see step 3d), the *error_type* is the type of error, and $auth\_code_4$ is generated using $DAK_{1,\,2}$. Note that since there was an error in the received message, the wrapped authentication key (*Group_Transaction_auth_key*) in the received message may not be correct, so $DAK_{1,\,2}$ is used as the authentication key.

   Center 1 may choose to resend the *translation request* information (not shown in the figure).

   Center 1 may notify Entity A of the problem in an error message (not shown in the figure).

5. (a) If the verification was successful and Entity B is a subscriber of Center 2, Center 2 translates the keying material by wrapping it in the KWK shared with Entity B ($KWK_{B,\,2}$):

   $$Entity\_B\ wrapped\_keys = \text{WRAP}(KWK_{B,\,2},\ KWK_{A,\,B},\ DAK_{A,\,B}\ ||\ Transaction\_auth\_key).$$

   (b) Center 2 then prepares and sends a message containing the *key transfer* information to Entity B.

key_transfer(*Entity_B_wrapped_keys*; *communicating_group: Entity_A*, *Entity_B;* $auth\_code_5$),

where $auth\_code_5$ is generated using *Transaction_auth_key*.

6. (a) Entity B unwraps the received keying material and attempts to verify $auth\_code_5$ using the transaction authentication key provided in the *key transfer* information (*Transaction_auth_key*). If the verification fails, or Entity B does not wish to establish a communicating group with Entity A, a message containing *error report* information is returned to Center 2:

   error_report(*previous_message_id*; *error_type*; $auth\_code_6$),

   where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 5b), the *error_type* is the type of error, and $auth\_code_6$ is generated using $DAK_{B,\,2}$. Note that when there is an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *key transfer* information may not be correct, so $DAK_{B,\,2}$ is used as the authentication key. Also, if Entity B does not want to establish a communicating group with Entity A, using the authentication key received in the *key transfer* information (*Transaction_auth_key*) may not be desirable.

   Center 2 may choose to resend the *key transfer* information if the previous message was in error (not shown in the figure).

   (b) If the verification is successful, and Entity B wishes to establish a communicating group with Entity A, Entity B sends a message containing *acknowledgement* information to Center 2:

   acknowledgement(*previous_msg_id*; $auth\_code_7$),

   where *previous_message_id* is the ID for the message containing the *key-transfer* information (see step 5b), and $auth\_code_7$ is generated using *Transaction_auth_key*.

7. If the acknowledgement is received correctly from Entity B, then the Center 2 forwards the *acknowledgement* information to Center 1, indicating that the communicating group has been established successfully:

   acknowledgement(*previous_msg_id*; $auth\_code_8$),

   where *previous_message_id* is the ID for the message containing the *translation request* information (see step 3d), and $auth\_code_8$ is generated using *Group_Transaction_auth_key*.

   For the sake of brevity, Center 1's receipt and verification of the message received from Center 2 containing the *acknowledgement* information is not discussed in detail here. However, if there is an error in the message, Center 1 could send a message to Center 2 containing *error report* information, and Center 2 could resend the *acknowledgement* information above.

8. (a) Assuming that the message containing the *acknowledgement* information is received correctly from Center 2, Center 1 wraps the keys for Entity A using the KWK shared with Entity A ($KWK_{A,1}$):

   $Entity\_A\_wrapped\_keys$ = WRAP($KWK_{A,\,1}$, $KWK_{A,\,B}$ || $DAK_{A,\,B}$ || $Transaction\_auth\_key$).

(b) Center 1 then prepares and sends a message containing the *key transfer* information to Entity A:

key_transfer(*Entity_A_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; $auth\_code_9$),

where $auth\_code_9$ is generated using *Transaction_auth_key*.

9. (a) Entity A attempts to verify $auth\_code_9$ using the transaction authentication key provided in the *key transfer* information (*Transaction_auth_key*). If the verification fails, a message containing *error report* information is returned to Center 1:

error_report(*previous_message_id*; *error_type*; $auth\_code_{10}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 8b), the *error_type* is the type of error, and $auth\_code_6$ is generated using $DAK_{A,\,1}$. Note that when there is an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *key transfer* information may not be correct, so $DAK_{A,\,1}$ is used as the authentication key.

Center 1 may choose to resend the *key transfer* information if the previous message was in error (not shown in the figure).

(b) If the message containing the *key transfer* information is received correctly from Center 1, then the Entity A sends the *acknowledgement* information to Center 1 indicating that the key transfer information was received correctly:

acknowledgement(*previous_msg_id*; $auth\_code_{11}$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 8b), and $auth\_code_{11}$ is generated using *Transactiun_auth_key*.

Note that for the sake of brevity, Center 1's receipt and handling of the *acknowledgement* information is not discussed in detail here. However, if the message containing the *acknowledgement* information cannot be verified, then Center 1 could send an error message to Entity A, and Entity A could resend the *acknowledgement* information above.

At this point, Entities A and B share a KWK and DAK as members of the same communicating group. However, only Entity A knows for sure that the keys are shared (because of the order that the keys were distributed by the multiple-center group in this example). Entity B could be notified of this fact by Entity A or B sending a cryptographically protected message to the other party (e.g., protected using the newly established KWK and DAK). Alternatively, a special purpose-confirmation message could be used to indicate successful establishment of the communicating group.

A.6 Using a Multiple-Center Group to Establish a Communicating Group Only Using its Key-Translation Services

Figure A.6 depicts an example of information flow for establishing a communicating group using the key-translation services of a multiple-center group. In this example, Center 1 and Center 2 are members of the same multiple-center group and share a KWK ($KWK_{1,\,2}$) and an authentication key ($DAK_{1,\,2}$). Entity A's agent to the group is Center 1; they share

$KWK_{A,\,1}$ and $DAK_{A,\,1}$. Entity B's agent to the group is Center 2; they share $KWK_{B,\,2}$ and $DAK_{B,\,2}$. Entity A and Center 1 can generate keys, but Center 2 cannot. Entities A and B wish to establish a communicating group.

Note that this example is very similar to the example in Appendix A.5; the main difference is that Entity A can generate keys.

In this example, Entity A generates keying material and sends it to its agent for translation for Entity B. Center 1 forwards the keying material to Center 2 for translation and providing the translated keying material to Entity B. After receiving an acknowledgement from Entity B that the keys have been received correctly, Center 2 sends an acknowledgement of correct receipt to Center 1, who forwards the acknowledgement to Entity A; the communicating group is then considered to be established.

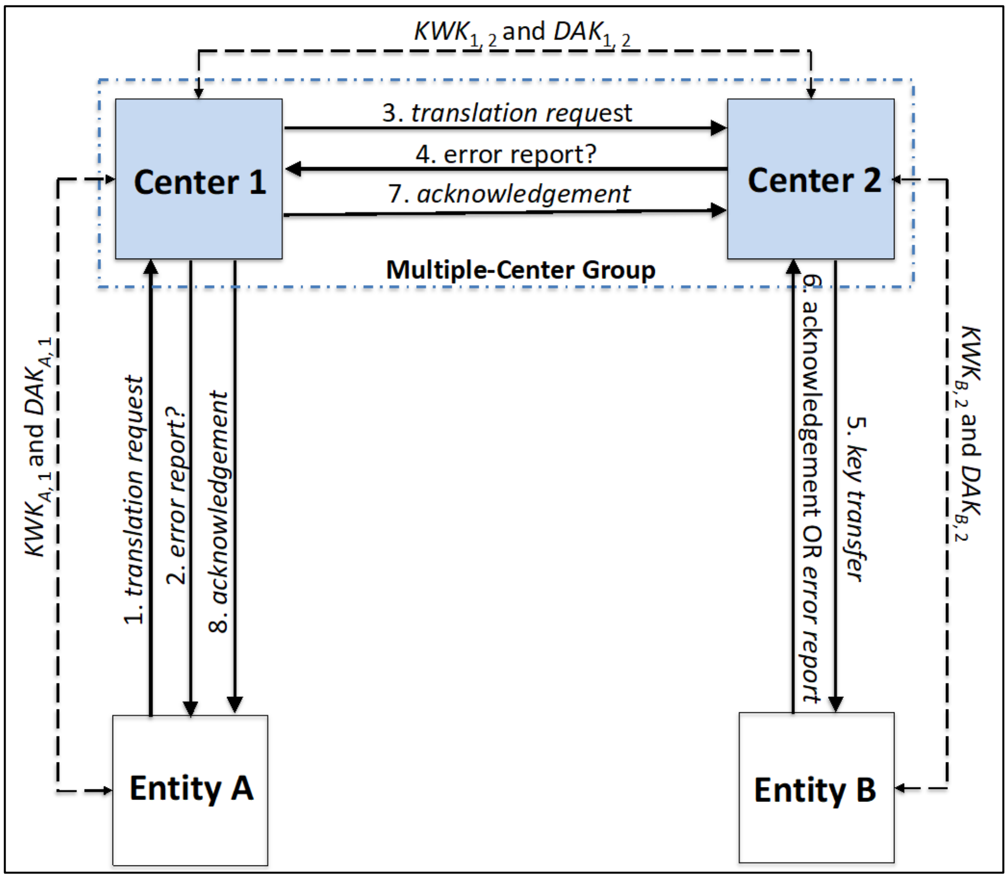

Figure A.6: Using the Translation Services of a Multiple-Center Group to Establish a Communicating Group

1. (a) Entity A generates a KWK, a DAK and a transaction authentication key (*Transaction_auth_key*) to be sent to Entity B.

   (b) Entity A wraps the keys in the KWK shared with its agent ($KWK_{A,\,1}$):

   $$wrapped\_keys = \text{WRAP}(KWK_{A,\,1},\, KWK_{A,\,B} \,\|\, DAK_{A,\,B} \,\|\, Transaction\_auth\_key).$$

(c) Entity A prepares and sends a message containing *translation request* information to Center 1 that includes the wrapped keys:

translation_request(*wrapped_keys*; *requester*: Entity_A; *communicating_group*: Entity_A, Entity B; $auth\_code_1$),

where $auth\_code_1$ is generated using the generated authentication key (*Transaction_auth_key*).

2. (a) Center 1 unwraps the wrapped keying material in the *translation request* information and uses the unwrapped transaction authentication key (*Transaction_auth*_key) to verify the message containing the *translation request* information.

   (b) If the verification fails, a message is sent to Entity A containing the *error report* information:

   error_report(*previous_message_id*; *error_type; auth_code₂*),

   where *previous_message_id* is the ID for the message containing the *translation request* information (see step 1c), the *error_type* is the type of error, and $auth\_code_2$ is computed using $DAK_{A,\,1}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *translation request* information may not be correct, so $DAK_{A,\,1}$ is used as the authentication key.

   Entity A may choose to resend the *translation request* information (not shown in the figure).

3. If the verification of the message containing the *key-generation request* is successful, but the other entity identified in the *key-generation request* information (Entity B) is not a subscriber of Center 1, then Center 1 suspects that Entity B may be a subscriber of another center in the multiple-center group (i.e., Center 2 in this example).

   (a) Center 1 needs to send *translation request* information to Center 2, so generates an authentication key (*Group_Transaction_auth_key*) and wraps it with the keys to be translated using the KWK shared with Center 2 ($KWK_{1,2}$).

   $$Center\_2\_wrapped\_keys = \text{WRAP}(KWK_{1,2}, KWK_{A,\,B} \parallel DAK_{A,\,B} \parallel Transaction\_auth\_key \parallel Group\_Transaction\_auth\_key).$$

   (b) Center 1 prepares and sends a message containing *translation request* information to Center 2 that includes the wrapped keys:

   translation_request(*Center_2_wrapped_keys*; *requester*: Center_1; *communicating_group*: Entity_A, Entity B; $auth\_code_3$),

   where $auth\_code_3$ is generated using *Group_Transaction_auth_key*.

4. (a) Center 2 unwraps the *translation request* information to obtain the authentication key (*Group_Transaction_auth_key*).

   (b) Center 2 attempts to verify $auth\_code_3$ using the unwrapped authentication key (*Group_Transaction_auth_key*). If the verification fails, or if Entity B is not a subscriber of Center 2, a message containing *error report* information is returned to Center 1, and the process is terminated.

error_report(*previous_message_id*; *error_type; auth_code*$_4$),

where *previous_message_id* is the ID for the message containing the *translation request* information (see step 3b), the *error_type* is the type of error, and *auth_code*$_4$ is computed on the message using $DAK_{1,\,2}$. Note that since there was an error in the received message, the wrapped authentication key (*Group_Transaction_auth_key*) in the *translation request* information may not be correct, so $DAK_{1,\,2}$ is used as the authentication key for the message containing the *error report* information.

Center 1 may choose to resend the *translation request* information or to notify Entity A of the problem in an error message (not shown in the figure).

5. (a) If the verification is successful, and Entity is a subscriber of Center 2, Center 2 translates the keys ($KWK_{A,\,B}$ || $DAK_{A,\,B}$ || *Transaction_auth_key*) by wrapping them in the KWK shared with Entity B ($KWK_{B,\,2}$):

   *Entity_B wrapped_key*s = WRAP($KWK_{B,\,2}$, $KWK_{A,\,B}$ || $DAK_{A,\,B}$ || *Transaction_auth_key*).

   (b) Center 2 then prepares and sends a message containing the *key transfer* information to Entity B.

   key_transfer(*Entity_B_wrapped_keys*; *communicating_group*: Entity_A, Entity_B; *auth_code*$_5$),

   where *auth_code*$_5$ is generated on the message using *Transaction_auth_key*.

6. (a) Entity B unwraps the *key transfer* information and attempts to verify *auth_code*$_5$ using the unwrapped authentication key (*Transaction_auth_key*).

   (b) If the verification fails, or Entity B does not want to establish a communicating group with Entity A, a message is sent to Center 2 containing *error report* inoformation.

   error_report(*previous_message_id*; *error_type; auth_code*$_6$),

   where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 5b), the *error_type* is the type of error, and *auth_code*$_6$ is computed using $DAK_{B,2}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*) in the *key transfer* information may not be correct, so $DAK_{B,\,2}$ is used as the authentication key for the message containing the *error report* information.

   Center 2 may choose to resend the *key transfer* information (not shown in the figure) (see step 5b).

   Alternatively, Center 2 could send a message containing *error report* information to Center 1 indicating that the keys could not be established between Entities A and B, authenticating the message using $DAK_{1,\,2}$; Center 1 could then forward the information to Entity A, authenticating the message using $DAK_{A,1}$. The transaction would then be considered as terminated. These messages are not shown in the figure.

   (c) If the verification is successful, and Entity B wants to establish a communicating group with Entity A, Entity B sends a message containing *acknowledgement* information to Center 2:

acknowledgement(*previous_msg_id*; $auth\_code_7$),

where *previous_message_id* is the ID for the message containing the *key transfer* information, and $auth\_code_7$ is generated using *Transaction_auth_key*.

7. If a message containing *acknowledgement* information is received correctly from Entity B, then Center 2 sends a message containing *acknowledgement* information to Center 1, indicating that the communicating group has been established successfully:

   acknowledgement(*previous_msg_id*; $auth\_code_8$),

   where *previous_message_id* is the ID for the message containing the *translation request* information (see step 3b), and $auth\_code_8$ is generated using *Group_Transaction_auth_key*.

8. If a message containing *acknowledgement* information is received correctly from Center 2, then Center 1 sends a message containing *acknowledgement* information to Entity A indicating that the communicating group has been established successfully:

   acknowledgement(*previous_msg_id*; $auth\_code_9$),

   where *previous_message_id* is the ID for the message containing the *key-generation request* information (see step 1c), and $auth\_code_9$ is generated using *Transaction_auth_key*.

At this point, Entities A and B share a KWK and DAK as members of the same communicating group. However, only Entity A knows for sure that the keys are shared. Entity B could be notified of this fact by Entity A or B sending a cryptographically protected message to the other party (e.g., protected using the newly established KWK and DAK). Alternatively, a special-purpose confirmation message could be used to indicate successful establishment of the communicating group.

A.7 Forwarding Keys Through an Intermediate Entity

Keying material can be forwarded to the ultimate recipient(s) through intermediate entities (see Figure A.7 for an example). In this example, a KDC shares a KWK and DAK with Entity A, and Entity A shares KWKs and DAKs as the Layer 1 keys with Entities B and C, i.e.,

- The KDC shares $KWK_{A,\,KDC}$ and $DAK_{A,\,KDC}$ with Entity A;
- Entity A shares $KWK_{A,\,B}$ and $DAK_{A,\,B}$ with Entity B; and
- Entity A shares $KWK_{A,\,C}$ and $DAK_{A,\,C}$ with Entity C.

In this example, the KDC generates keying material (e.g., an AEK) to be shared by Entities B and C and distributes it via one of its subscribers (Entity A). Entities B and C become a communicating group, but since they do not share a KWK, they cannot generate further keys without the assistance of the KDC. Although Entity A is privy to the keys (since it assisted in their distribution), Entity A is not intended to be part of that communicating group for this example.

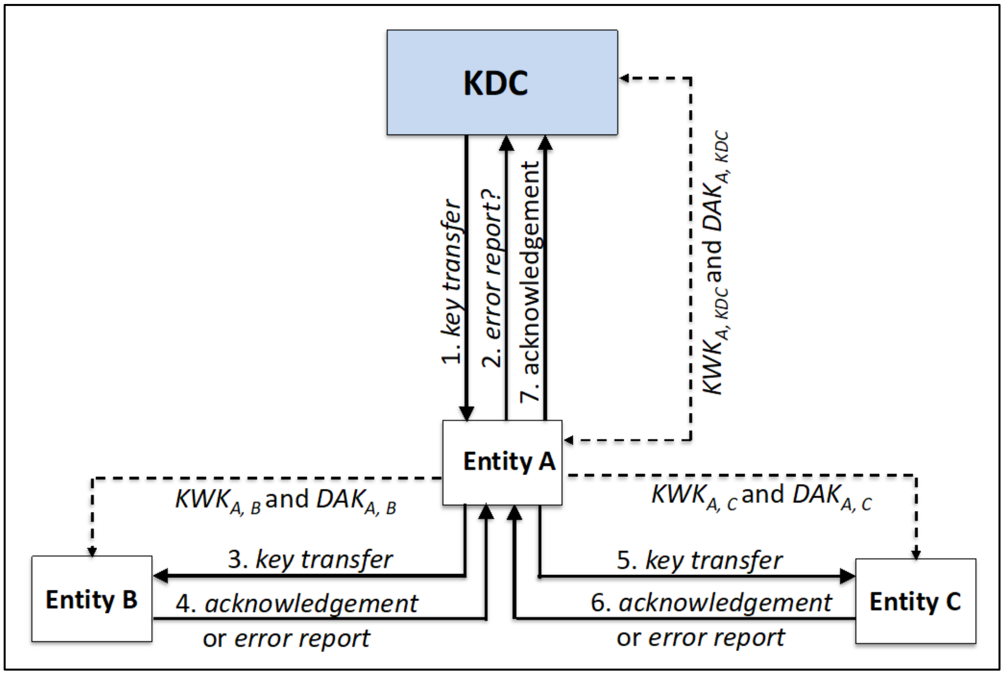

Figure A.7: *key transfer* through an Intermediate Entity

1. (a) The KDC generates an AEK and authentication keys (*Transaction_auth_key*$_1$ and *Transaction_auth_key*$_2$) to be used for message authentication and wraps them for Entity A.

wrapped_keys = WRAP($KWK_{A,\ KDC}$, $AEK_{B,C}$ || *Transaction_auth_key*$_1$ || *Transaction_auth_key*$_2$).

(b) The KDC prepares and sends a message containing *key transfer* information to Entity A:

key_transfer(*wrapped_keys*; *communicating_group*: Entity B, Entity C; *auth_code*$_1$),

where *auth_code*$_1$ is computed on the message containing the *key transfer* information using *Transaction_auth_key*$_2$.

2. Entity A unwraps the *key transfer* information using the KWK shared with the KDC ($KWK_{A,\ KDC}$) and attempts to verify the received message using *Transaction_auth_key*$_2$.

(a) If the verification fails, an error message is sent to the KDC containing *error report* information:

error_report(*previous_message_id*; *error_type*; *auth_code*$_2$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 1b), the *error_type* is the type of error, and *auth_code*$_2$ is generated using $DAK_{A,\ KDC}$. Note that since there was an error in the received message, the wrapped authentication key (*Transaction_auth_key*$_2$) in the message may not be correct, so $DAK_{A,\ KDC}$ is used as the authentication key.

The KDC may choose to resend the *key transfer* information (not shown in the figure).

Steps 3 and 5 (these steps are combined to avoid repetitious descriptions):

(a) If the verification is successful, the wrapped keys destined for Entities B and C are extracted and wrapped for each intended recipient:

$$Entity\_B\_wrapped\_keys = \text{WRAP}(KWK_{A,\,B}, AEK_{B,C} \parallel Transaction\_auth\_key_1).$$

$$Entity\_C\_wrapped\_keys = \text{WRAP}(KWK_{A,\,C}, AEK_{B,C} \parallel Transaction\_auth\_key_1).$$

(b) The appropriate wrapped keys are placed in *key transfer* messages for each recipient, and an authentication code is computed for each message ($auth\_code_3$ and $auth\_code_4$) using the appropriate transaction authentication key ($Transaction\_auth\_key_1$):

key_transfer(*Entity_B_wrapped_keys*; *communicating_group*: Entity_B, Entity_C; $auth\_code_3$) is sent to Entity B.

key_transfer(*Entity_C_wrapped_keys*; *communicating_group*: Entity_B, Entity_C; $auth\_code_4$) is sent to Entity B.

Steps 4 and 6:

(a) Entities B and C unwrap the keys received in the *key transfer* information of their respective messages and attempt to verify the authentication codes ($auth\_code_3$ and $auth\_code_4$, respectively) using $Transaction\_auth\_key_1$.

(b) If the verification fails, or the receiving entity does not want to be a member of the communicating group, a message is sent to Entity A containing *error report* information:

Entity B would send ***error_report***(*previous_message_id*; *error_type*; $auth\_code_5$)

Entity C would send ***error_report***(*previous_message_id*; *error_type*; $auth\_code_6$)

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 3/5 b), and the *error_type* is the type of error. Entity B would generate $auth\_code_5$ using $DAK_{A,\,B}$; Entity C would generate $auth\_code_6$ using $DAK_{A,\,C}$. Note that since there was an error in the received message, the wrapped authentication key ($Transaction\_auth\_key_1$) in the *key transfer* information may not be correct, so $DAK_{A,\,B}$ and $DAK_{A,\,C}$ would be used as the authentication keys.

Entity A may choose to resend the *key transfer* information (not shown in the figure).

(c) If the verification is successful, and both entities want to establish a communicating group with each other, a message containing *acknowledgement* information is sent to Entity A:

Entity B would send ***acknowledgement***(*previous_msg_id*; $auth\_code_7$)

Entity C would send ***acknowledgement***(*previous_msg_id*; $auth\_code_8$),

where *previous_message_id* is the ID for the message containing the *key transfer* information (see step 3/5 b), and $auth\_code_7$ and $auth\_code_8$ are generated using $Transaction\_auth\_key_1$.

7. If the message containing the *acknowledgement* information is received correctly from both Entities B and C, then Entity A sends a message to the KDC containing *acknowledgement* information indicating that the communicating group has been established successfully:

acknowledgement(*previous_msg_id*; $auth\_code_9$),

where *previous_message_id* is the ID for the message containing the *key-transfer* information (see step 1b), and $auth\_code_9$ is generated using $Transaction\_auth\_key_2$.

At this point, Entities B and C share an AEK as members of the same communicating group. However, only Entity A and the KDC know for sure that the keys are shared. Enitites B and C could be notified of this fact in a couple of ways:

- By Entity B or C sending a cryptographically protected message to the other party (e.g., protected using the newly established AEK); or
- By Entity A sending *acknowledgement* information to Entities B and C indicating that the establishment of the communicating group has been completed (not shown in the figure).

A.8 Requesting Key Revocation and Confirmation

A.8.1 Example 1

Figure A.8a is an example of using a *revocation request* and corresponding *revocation confirmation*. In this example, a KDC sends a revocation request to the members of a communicating group (Entities A and B) to terminate the group by revoking the Level 1 key in their key hierarchy ($KWK_{A,\,B}$); presumably, the KDC was a participant in establishing that key. Entity A shares $KWK_{A,\,KDC}$ and $DAK_{A,\,KDC}$ with the KDC; Entity B shares $KWK_{B,\,KDC}$ and $DAK_{B,\,KDC}$ with the KDC.

The keys shared by Entities A and B consist of a Layer 1 key ($KWK_{A,B}$) and a layer 2 $DAK_{A,\,B}$, which were established previously using the KDC (see Appendix A.3), and several lower-layer keys established within the communicating group (i.e., Entities A and B) using $KWK_{A,\,B}$ after the group was established (see Appendix A.1 for the process):

- $KWK_{A,\,B}$ was used to wrap KWK_{Layer_2} and DAK_{Layer_2}.
- KWK_{Layer_2} was used to wrap KDK_{Layer_3}, and
- KDK_{Layer_3} was used to derive DEK_{Layer_4} and DAK_{Layer_4}.

In this example, the revocation request is sent directly to each entity by the KDC so that each will acknowledge that they have fulfilled the request. Note that in this example, both revocation requests are sent before expecting the return of the corresponding *revocation confirmation* or *error report* information. This is a design decision for this example (not a requirement) to allow each entity to find and destroy all copies of keys affected by the *revocation request* information (i.e., all keys lower in the key hierarchy).

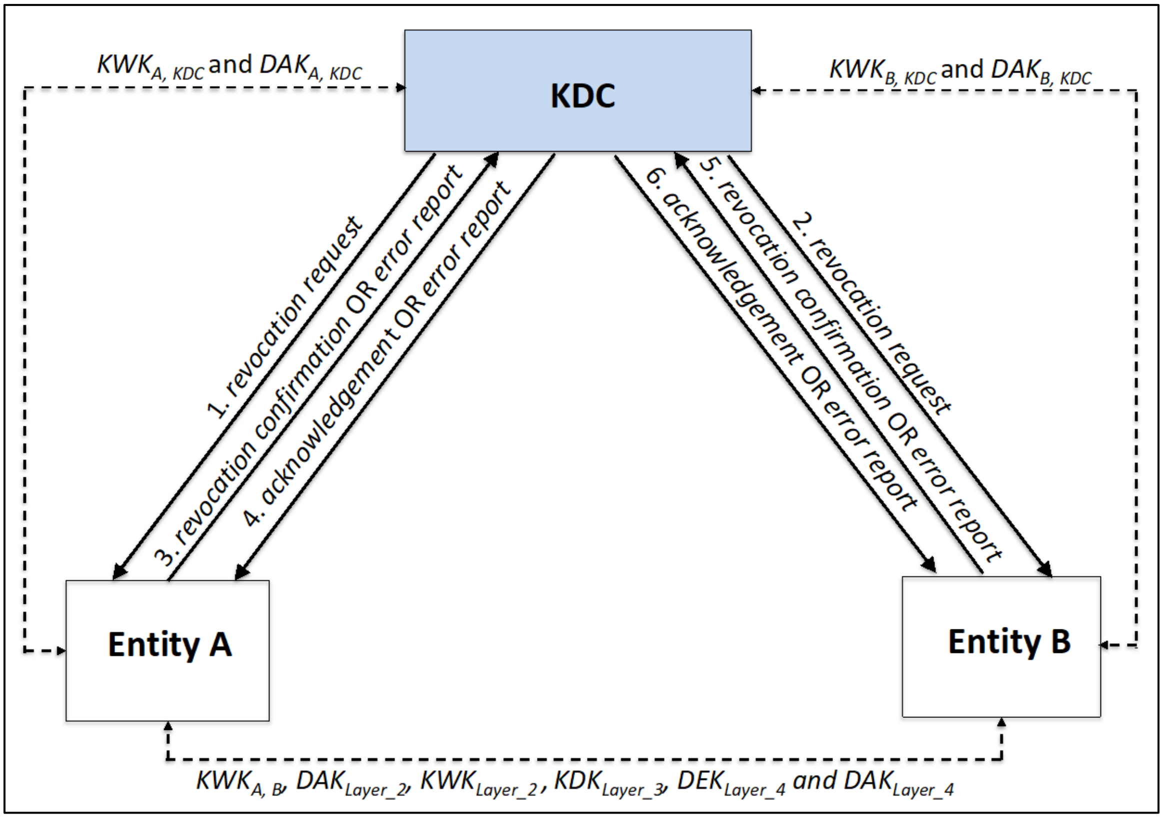

Figure A.8a: KDC Revocation of a Communicating Group

1. (a) The KDC generates an authentication key (*Transaction_auth_key*$_A$) for the message containing the *revocation request* information to be sent to Entity A and wraps it using the KWK shared with Entity A (*KWK*$_{A,\ KDC}$):

   wrapped_auth_key_A = WRAP(*KWK*$_{A,\ KDC}$, *Transaction_auth_key*$_A$).

   (b) The KDC prepares and sends a message to Entity A containing *revocation request* information that requests that Entity A revoke the Level 1 KWK (*KWK*$_{A,\ B}$) shared with Entity B and all keys beneath it in the key hierarchy:

   revocation_request(ID of *KWK*$_{A,\ B}$; *wrapped_auth_key_A*; *auth_code*$_1$),

   where *auth_code*$_1$ is generated on the message using *Transaction_auth_key*$_A$.

2. (a) Likewise, the KDC generates an authentication key (*Transaction_auth_key*$_B$) for the message containing the *revocation request* information to be sent to Entity B and wraps it using the KWK shared with Entity B (*KWK*$_{B,\ KDC}$):

   wrapped_auth_key_B = WRAP(*KWK*$_{B,\ KDC}$, *Transaction_auth_key*$_B$).

   (b) The KDC prepares and sends a message to Entity B containing the *revocation request* information that requests that Entity B revoke the Level 1 KWK (*KWK*$_{A,\ B}$) shared with Entity A:

   revocation_request(ID_of_*KWK*$_{A,\ B}$; *wrapped_auth_key_B*; *auth_code*$_2$),

   where *auth_code*$_2$ is generated on the message using *Transaction_auth_key*$_B$.

3. (a) Entity A unwraps the authentication key and attempts to verify the received message. If the verification fails, a message containing *error report* information is sent to the KDC and the process is terminated:

error_report(*previous_message_id*; *error_type; auth_code*$_3$),

where *previous_message_id* is the ID for the message containing the *revocation request* information (see step 1b), the *error_type* is the type of error, and *auth_code*$_3$ is computed using $DAK_{A,\,KDC}$. Note that since there was an error in the received message, the wrapped authentication key ($Transaction\_auth\_key_A$) in the message may not be correct, so $DAK_{A,\,KDC}$ is used as the authentication key.

The KDC would most likely resend the message, in this case.

(b) If the verification is successful, Entity A destroys all copies of $KWK_{A,\,B}$ and any keys lower in the key hierarchy (i.e., KWK_{Layer_2}, DAK_{Layer_2}, KDK_{Layer_3}, DEK_{Layer_4} and DAK_{Layer_4}).

(c) Entity A prepares and sends a message containing *revocation confirmation* information to the KDC:

revocation_confirmation(ID_of_$KWK_{A,\,B}$; *auth_code*$_4$),

where *auth_code*$_4$ is computed on the message using $Transaction\_auth\_key_A$.

4. The KDC attempts to verify *auth_code*$_4$ using the authentication key used for the message containing the *revocation request* information (see step 1b) (i.e., $Transaction\_auth\_key_A$).

(a) If the verification fails, a message containing *error report* information is returned to Entity A, and the process is terminated.

error_report(*previous_message_id*; *error_type; auth_code*$_5$),

where *previous_message_id* is the ID for the message containing the *revocation confirmation* information (see step 3c), the *error_type* is the type of error, and *auth_code*$_5$ is computed on the message using $Transaction\_auth\_key_A$. Since the *revocation request* was received correctly, $Transaction\_auth\_key_A$ can be used.

Entity A may choose to resend the message (not shown in the figure).

(b) If the verification is successful, the KDC sends a message containing *acknowledgement* information to Entity A:

acknowledgement(*previous_msg_id*; *auth_code*$_6$),

where *previous_message_id* is the ID for the message containing the *revocation confirmation* information (see step 3c), and *auth_code*$_6$ is generated on the message using $Transaction\_auth\_key_A$.

5. (a) Entity B unwraps the authentication key and attempts to verify the received message. If the verification fails, a message containing *error report* information is sent to the KDC and the process is terminated:

error_report(*previous_message_id*; *error_type; auth_code*$_7$), where *previous_message_id* is the ID for the message containing the *revocation request* information (see step 2b), the *error_type* is the type of error, and $auth\_code_7$ is computed on the message using $DAK_{B,KDC}$. Note that since there was an error in the received message, the wrapped authentication key ($Transaction\_auth\_key_B$) in the message may not be correct, so $DAK_{B,KDC}$ is used as the authentication key.

The KDC would most likely resend the message, in this case.

(b) If the verification is successful, Entity B destroys all copies of $KWK_{A,B}$ and any keys lower in the key hierarchy (i.e., KWK_{Layer_2}, DAK_{Layer_2}, KDK_{Layer_3}, DEK_{Layer_4} and DAK_{Layer_4}).

(c) Entity B prepares and sends a message containing *revocation confirmation* information to the KDC:

revocation_confirmation(ID_of_$KWK_{A,B}$; $auth\_code_8$),

where $auth\_code_8$ is computed on the message using $Transaction\_auth\_key_B$.

6. The KDC attempts to verify $auth\_code_8$ using the authentication key used for the message containing the *revocation request* information (see step 2b) (i.e., $Transaction\_auth\_key_B$).

(a) If the verification fails, a message containing the *error report* information is returned to Entity B, and the process is terminated.

error_report(*previous_message_id*; *error_type*; $auth\_code_9$),

where *previous_message_id* is the ID for the message containing the *revocation confirmation* information (see step 5c), the *error_type* is the type of error, and $auth\_code_9$ is computed on the message using $Transaction\_auth\_key_B$. Since the *revocation request* was received correctly, $Transaction\_auth\_key_B$ can be used.

Entity B may choose to resend the message (not shown in the figure).

(b) If the verification is successful, the KDC sends a message containing *acknowledgement* information to Entity B:

acknowledgement(*previous_msg_id*; $auth\_code_{10}$),

where *previous_message_id* is the ID for the message containing the *revocation confirmation* information (see step 5c), and $auth\_code_{10}$ is generated on the message using $Transaction\_auth\_key_B$.

A.8.2 Example 2

In this example, a communicating group consists of Entities A and B, with shared keys shown in Figure A.8b. Entity A wishes to revoke the KDK and all keys below it in the key hierarchy (e.g., because the KDK has been compromised or has been used too many times to derive keys).

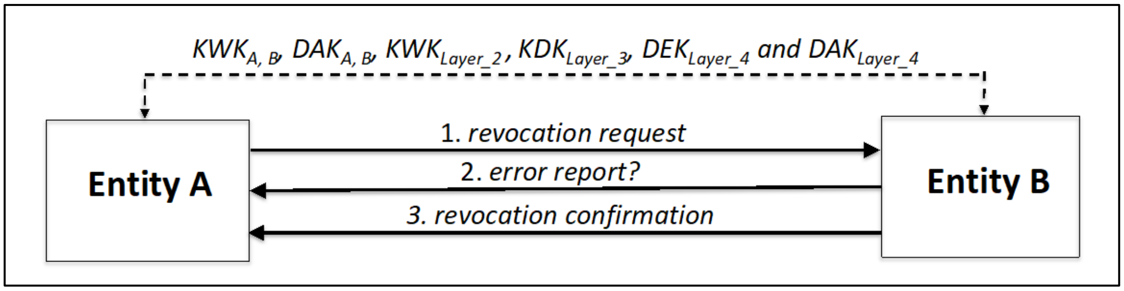

Figure A.8b: Revocation of Lower-level Keys

1. (a) If Entity A has a key-generation capability:

    - Entity A generates an authentication key (*Transaction_auth_key*) for the message containing the *revocation request* information to be sent to Entity B and wraps it using the KWK shared with Entity B ($KWK_{A,\,B}$):

      wrapped_auth_key = WRAP($KWK_{A,\,B}$, *Transaction_auth_key*).

    - Entity A prepares and sends a message to Entity B containing *revocation request* information that requests that Entity B revoke the KDK shared with Entity A and all keys beneath it in the key hierarchy:

      revocation_request(ID_of_KDK_{Layer_3}; *wrapped_auth_key*; $auth\_code_1$),

      where $auth\_code_1$ is generated on the message using *Transaction_auth_key*.

    (b) If Entity A does not have a key-generation capability:

    - Entity A will use $DAK_{A,\,B}$ as the authentication key for the message containing the *revocation request* information. Let "DAKAB" be the name of that key.

    - Entity A prepares and sends a message to Entity B containing *revocation request* information that requests that Entity B revoke the KDK shared with Entity A and all keys beneath it in the key hierarchy:

      revocation_request(ID_of_KDK_{Layer_3}; DAKAB ; $auth\_code_2$),

      where $auth\_code_2$ is generated on the message using $DAK_{A,\,B}$.

2. (a) If a wrapped key is included in the *revocation request* information: Entity B unwraps the authentication key using $KWK_{A,\,B}$, obtaining *Transaction_auth_key*.

    (b) If the ID of an authentication key is included in the revocation request information, that key is used as the authentication key (i.e., $DAK_{A,\,B}$, in this case).

    (c) Entity B attempts to verify the received message. If the verification fails, a message containing *error report* information is sent to Entity A, and the process is terminated:

      error_report(*previous_message_id*; *error_type;* $auth\_code_3$),

    where *previous_message_id* is the ID for the message containing the *revocation request* information (see step 1), the *error_type* is the type of error, and $auth\_code_3$ is computed using $DAK_{A,\,B}$. Note that since there was an error in the received message, $DAK_{A,\,B}$ is used as the authentication key.

Entity A would most likely resend the message, in this case.

3. (a) If the verification is successful, Entity B destroys all copies of KDK_{Layer_3} and any keys lower in the key hierarchy (i.e., DEK_{Layer_4} and DAK_{Layer_4}).

   (b) Entity B prepares and sends a message containing *revocation confirmation* information to Entity A:

   revocation_confirmation(ID_of_ KDK_{Layer_3}; $auth\_code_4$),

   where $auth\_code_4$ is computed on the message using the authenication key usd for the message containing the *revocation request* information (i.e., either *Transaction_auth_key* or $DAK_{A,\,B}$).

Appendix B: References

[DSKPP] *Dynamic Symmetric Key Provisioning Protocol (DSKPP)*; RFC 6063; Doherty, Pei, Machani, and Nystrom; Internet Engineering Task Force, December 2010.

https://tools.ietf.org/html/rfc6063

[FIPS140-2] *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards (FIPS) Publication FIPS 140-2, U.S. Department of Commerce/NIST, December 3, 2002.

https://doi.org/10.6028/NIST.FIPS.140-2

[FIPS 180-4] *The Secure Hash Standard*, Federal Information Processing Standards (FIPS) Publication FIPS-180-4, U. S. Department of Commerce/NIST, August 4, 2015.

https://doi.org/10.6028/NIST.FIPS.186-4

[FIPS-197] *Advanced Encryption Standard (AES)*, Federal Information Processing Standards (FIPS) Publication FIPS 197, U. S. Department of Commerce/NIST, November 26, 2001.

https://doi.org/10.6028/NIST.FIPS.197

[FIPS 198-1] *The Keyed-Hash Message Authentication Code (HMAC)*, Federal Information Processing Standards (FIPS) Publication FIPS 198-1, U. S. Department of Commerce/NIST, July 2008.

https://doi.org/10.6028/NIST.FIPS.198-1

[FIPS 199] *Standards for Security Categorization of Federal Information and Information Systems*, Federal Information Processing Standards (FIPS) Publication FIPS 199, U. S. Department of Commerce/NIST, March 2006.

https://doi.org/10.6028/NIST.FIPS.199

[FIPS 200] *Minimum Security Requirements for Federal Information and Information Systems*, , Federal Information Processing Standards (FIPS) Publication FIPS 200, U. S. Department of Commerce/NIST, February 2004.

https://doi.org/10.6028/NIST.FIPS.200

[FIPS 202] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*, Federal Information Processing Standards (FIPS) Publication FIPS-202, U. S. Department of Commerce/NIST, August 4, 2015.

https://doi.org/10.6028/NIST.FIPS.202

[Kerberos] *Kerberos: The Network Authentication Protocol*, Massachusetts Institute of Technology, September 25, 2017

https://web.mit.edu/kerberos/

[KM in WSN] "Key Management in Wireless Sensor Networks;" Mansour, Chalhoub, and Lafourcade; *Journal of Sensor and Actuator Networks*, ISSN 2224-2708; September 7, 2015.

http://www.mdpi.com/2224-2708/4/3/251

[NISTIR 8105] *Report on Post-Quantum Cryptography*; Chen, Jordan, Liu, Moody, Peralta, Perlner, and Smith-Tone; National Institute of Standards and Technology, April 2016.

https://doi.org/10.6028/NIST.IR.8105

[NISTIR 8114] *Report on Lightweight Cryptography*; NISTIR 8114; McKay, Bassham, Turan, and Mouha; National Institute of Standards and Technology, March 2017.

https://doi.org/10.6028/NIST.IR.8114

[RFC 4107] *Guidelines for Cryptographic Key Management*, RFC 4107, Bellovin and Housley, The Internet Society, June 2005.

https://tools.ietf.org/html/rfc4107

[S/MIME] *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification*, RFC 5751, Ramsdell and Turner, The Internet Society, January 2010.

https://tools.ietf.org/html/rfc5751

The IETF LAMPS working group (see https://tools.ietf.org/wg/lamps/) has been developing a replacement for RFC 5751; the latest draft is available at https://tools.ietf.org/wg/lamps/draft-ietf-lamps-rfc5751-bis/.

[SP 800-38A] *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*, SP 800-38A, M. Dworkin, National Institute of Standards and Technology, December 2001.

https://doi.org/10.6028/NIST.SP.800-38A

[SP800-38B] *Recommendation for Block Cipher Modes of Operation: the CMAC Authentication Mode for Authentication*, SP 800-38B, M. Dworkin, National Institute of Standards and Technology, October, 2016.

https://doi.org/10.6028/NIST.SP.800-38B

[SP 800-38C] *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*, SP 800-38C, M. Dworkin, National Institute of Standards and Technology, May 2004.

https://doi.org/10.6028/NIST.SP.800-38C

[SP 800-38D] *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, SP 800-38D, M. Dworkin, National Institute of Standards and Technology, November 2007.

https://doi.org/10.6028/NIST.SP.800-38D

[SP800-38E] *Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices*, SP 800-38E, M. Dworkin, January 2010.

https://doi.org/10.6028/NIST.SP.800-38E

[SP 800-38F] *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*, SP 800-38F, M. Dworkin, National Institute of Standards and Technology, December 2012.

https://doi.org/10.6028/NIST.SP.800-38F

[SP 800-38G] *Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption*, M. Dworkin, National Institute of Standards and Technology, March 2016.

https://doi.org/10.6028/NIST.SP.800-38G

[SP 800-56A] *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*; SP 800-56A, Revision 2; E. Barker, L. Chen, A. Roginsky, and M. Smid; May 2010.

https://doi.org/10.6028/NIST.SP.800-56Ar3

[SP 800-56B] *Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography*; SP 800-56B, Revision 1; E. Barker, L. Chen, and D. Moody; September 2014.

https://doi.org/10.6028/NIST.SP.800-56Br1

[SP800-57 Pt. 1] *Recommendation for Key Management: Part 1: General*, Special Publication 800-57 Part 1, Revision 4, E. Barker, National Institute of Standards and Technology, January 2016.

https://doi.org/10.6028/NIST.SP.800-57pt1r4

[SP 800-57 Pt. 2] *Recommendation for Key Management: Part 2: Best Practices for Key Management Organizations*; Special Publication 800-57 Part 2, Revision 1 DRAFT, E. Barker, and W. Barker; National Institute of Standards and Technology; April 2018.

https://csrc.nist.gov/CSRC/media/Publications/sp/800-57-part-2/rev-1/draft/documents/sp800-57pt2-r1-draft.pdf

[SP 800-57 Pt. 3] *Recommendation for Key Management: Part 3: Application-Specific Key Management Guidance*, Special Publication 800-57 Part 3, E. Barker and Dang, National Institute of Standards and Technology January 2015.

https://doi.org/10.6028/NIST.SP.800-57pt3r1

[SP 800-88] *Guidelines for Media Sanitization*; Special Publication 800-88; R. Kissel, M. Scholl, S. Skolochenko, and X. Li; National Institute of Standards and Technology; September 2006.

https://doi.org/10.6028/NIST.SP.800-88r1

[SP 800-90A] *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*, SP 800-90A, Revision 1, E. Barker and J. Kelsey, National Institute of Standards and Technology; June 2015.

https://doi.org/10.6028/NIST.SP.800-90Ar1

[SP 800-108] *Recommendation for Key Derivation Using Pseudorandom Functions (Revised)*, Special Publication 800-108, L. Chen, National Institute of Standards and Technology, October 2009.

https://doi.org/10.6028/NIST.SP.800-108

[SP 800-131A] *Transitions: Recommendation for the Use of Cryptographic Algorithms and Key Lengths*, NIST SP 800-131A, Revision 1, E. Barker and Q. Dang, November 2015.

https://doi.org/10.6028/NIST.SP.800-131Ar1

[SP 800-152] *A Profile for U.S. Federal Cryptographic Key Management Systems (CKMS)*; NIST SP 800-152; E. Barker, Smid, and Branstad; National Institute of Standards and Technology; October 2015

https://doi.org/10.6028/NIST.SP.800-152

[SP 800-175B] Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, E. Barker, National Institute of Standards and Technology, August 2016.

https://doi.org/10.6028/NIST.SP.800-175B

[SP 800-185] *SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash*; NIST SP 800-185; Kelsey, Chang, and Perlner; National Institute of Standards and Technology; December 2016.

https://doi.org/10.6028/NIST.SP.800-185

[X9.17] American National Standard X9.17, Financial Institution Key Management (Wholesale), April 1985, Withdrawn.

[X9.28] American National Standard X9.28, Financial Institution Multiple Center Key Management (Wholesale), June 1991, Withdrawn.