

Superseded

**Cable Data Services
DOCSIS® Provisioning of EPON Specifications**

DPoE™ Security and Certificate Specification

DPoE-SP-SECv1.0-I01-110225

ISSUED

Notice

This DPoE specification is the result of a cooperative effort undertaken by certain member companies of Cable Television Laboratories, Inc. for the benefit of the cable industry and its customers. This document may contain references to other documents not owned or controlled by CableLabs. Use and understanding of this document may require access to such other documents. Designing, manufacturing, distributing, using, selling, or servicing products, or providing services, based on this document may require intellectual property licenses from third parties for technology referenced in this document.

Copied and Distributed by CableLabs under License.

Superseded

Document Status Sheet

Document Control Number:	DPoE-SP-SECv1.0-I01-110225			
Document Title:	DPoE™ Security and Certificate Specification			
Revision History:	I01 - Released 02/25/11			
Date:	February 25, 2011			
Status:	Work in Progress	Draft	Issued	Closed
Distribution Restrictions:	Author Only	CL/Member	CL/Member/Vendor	Public

Key to Document Status Codes

Work in Progress	An incomplete document, designed to guide discussion and generate feedback that may include several alternative requirements for consideration.
Draft	A document in specification format considered largely complete, but lacking review by Members and vendors. Drafts are susceptible to substantial change during the review process.
Issued	A stable document, which has undergone rigorous member and vendor review and is suitable for product design and development, cross-vendor interoperability, and for certification testing.
Closed	A static document, reviewed, tested, validated, and closed to further engineering change requests to the specification through CableLabs.

Trademarks

Advanced Digital Cable™, CableCARD™, CableHome®, CableLabs®, CableNET®, CableOffice™, CablePC™, DCAS™, DOCSIS®, DPoE™, EBIF™, eDOCSIS™, EuroDOCSIS™, EuroPacketCable™, Go2BroadbandSM, M-Card™, M-CMTS™, OCAP™, OpenCable™, PacketCable™, PCMM™, and tru2way® are marks of Cable Television Laboratories, Inc. All other marks are the property of their respective owners.

Superseded

Contents

1	INTRODUCTION	1
1.1	DPoE Introduction.....	1
1.2	Scope	2
1.3	Goals.....	2
1.4	Requirements	2
1.5	Organization of Specifications	3
1.6	DPoE Specifications.....	3
1.7	Reference Architecture	4
1.8	DPoE Interfaces and Reference Points	5
2	REFERENCES	8
2.1	Normative References	8
2.2	Informative References.....	9
2.3	Reference Acquisition	9
3	TERMS AND DEFINITIONS.....	10
3.1	DPoE Elements.....	10
3.2	Other Terms.....	11
4	ABBREVIATIONS AND ACRONYMS.....	12
5	OVERVIEW	14
5.1	Subscriber Data Privacy	14
5.1.1	Unicast Traffic encryption.....	14
5.1.2	Unicast traffic Key exchange algorithm.....	15
5.1.3	Multicast Traffic encryption	15
5.2	Service Provider Network Security	15
5.2.1	Control Message Encryption	15
5.2.2	IP Denial of Service attack mitigation.....	15
5.2.3	Ethernet Denial of Services Mitigation: Broadcast MAC forwarding	15
5.2.4	Limitation of the MAC address learning capacity.....	16
5.2.5	MAC address binding	16
5.3	eDOCSIS	16
6	ENCRYPTED FRAME FORMAT	17
6.1	1G Downstream-only Cipher Suite (1Down).....	17
6.2	10G Zero-Overhead Cipher Suite (10Down, 10Bi).....	17
6.2.1	10G Zero-Overhead Frame Format	17
7	KEY MANAGEMENT PROTOCOLS	19
7.1	1G Downstream-only Key Exchange Protocol.....	19
7.1.1	Set Key Exchange Timer.....	20
7.1.2	Create Key Exchange Timer	20
7.1.3	Generate Random Key.....	20
7.1.4	Send Key to DPoE System	20
7.1.5	Create Switchover Verification Timer (Optional)	21
7.1.6	Switchover Verification Event (Optional for ONU).....	22
7.1.7	Key Exchange Timer Expiration Event.....	22
7.1.8	Link Deregistration Event.....	22
7.1.9	Detecting Key Exchange Failures at the DPoE System.....	23
7.1.10	DPoE OAM Key Exchange Messages	24

Superseded

7.1.11	Summary of Down Encryption Parameters.....	25
7.2	10G Downstream-only Key Exchange Protocol.....	25
7.3	10G Bidirectional Key Exchange Protocol.....	25
8	AUTHENTICATION AND ENCRYPTION	26
8.1	DOCSIS and DPoE Authentication Comparison.....	26
8.2	DPoE ONU authentication	27
8.2.1	DPoE ONU MAC Address Identity.....	27
8.2.2	DPoE ONU Authentication.....	28
8.3	Use of EAP-TLS for DPoE ONU Authentication	28
8.3.1	EAPOL Notes.....	30
8.3.2	EAP Notes.....	30
8.3.3	TLS Notes.....	30
9	SECURE PROVISIONING.....	31
9.1	ONU and CM Management Comparison	31
10	USING CRYPTOGRAPHIC KEYS.....	32
10.1	DPoE System.....	32
10.2	DPoE ONU	32
10.3	Authentication of Dynamic Service Requests	32
11	CRYPTOGRAPHIC METHODS	33
11.1	General encryption requirements.....	33
11.2	DPoE Cipher Suites	34
11.2.1	1G downstream-only Cipher Suite (1Down)	34
11.2.2	10G Downstream-only Cipher Suite (10Down).....	34
11.2.3	10G Bidirectional Cipher Suite (10Bi).....	34
11.3	1 Down Cryptographic Method	34
11.3.1	Encrypting Frames	34
11.3.2	Decrypting Frames	35
11.3.3	Initial Vector (IV) Source	35
11.4	10 Down / 10Bi Cryptographic Method	36
11.4.1	Encrypting Frames	36
11.4.2	Decrypting Frames	37
11.4.3	Initial Vector (IV) Source	37
11.4.4	MPCP Jitter Correction.....	37
12	PHYSICAL PROTECTION OF SECURITY DATA IN THE ONU.....	39
13	X.509 CERTIFICATE PROFILE AND MANAGEMENT	40
13.1	X.509 Certificate Verification	40
13.2	Certificate Transport and Verification.....	40
14	SECURE SOFTWARE DOWNLOAD (SSD).....	42
14.1	Secure File Transfer across the D Interface.....	42
14.2	Secure File Transfer across the TU Interface	42
APPENDIX I	EXAMPLE FRAMES	43
I.1	AES 128 CFB Encrypted Frame.....	43
APPENDIX II	REFERENCE AES IMPLEMENTATION (C PROGRAMMING LANGUAGE).....	44
APPENDIX III	ACKNOWLEDGMENTS.....	68

Superseded

Figures

Figure 1 - DPoE Reference Architecture.....	5
Figure 2 - DPoE Interfaces and Reference Points	6
Figure 3 - DPoE Elements	10
Figure 4 - Cipher Text Region (1Down)	17
Figure 5 - Security Octet (1Down)	17
Figure 6 - Security Octet (10G Zero Overhead).....	18
Figure 7 - DPoE ONU Key Exchange State Machine.....	19
Figure 8 - Setting the Key Exchange Timer	20
Figure 9 - Send Key to the DPoE System	21
Figure 10 - Key Exchange Failure Detected by Link.....	22
Figure 11 - Changing Keys On Timer Expiration	23
Figure 12 - Key Exchange Failure Detected By DPoE System.....	24
Figure 13 - Set Key Exchange Timer Failure Detected by DPoE System.....	24
Figure 14 - Authentication in Bidirectional Methods.....	26
Figure 15 - Authentication in Downstream-Only Methods.....	27
Figure 16 - EAP-TLS Message Sequence	29
Figure 17 - EAP-TLS Frame Format.....	29
Figure 18 - Encrypting Data with Cipher Feedback Algorithm	34
Figure 19 - Decrypting Data with Cipher Feedback Algorithm	35
Figure 20 - Determining the Initial Vector	35
Figure 21 - Octet order within the Initial Vector (1G)	36
Figure 22 - Encrypting Data with CTR Mode	36
Figure 23 - Octet order within the Initial Vector (10G).....	37

Tables

Table 1 - DPoE Specifications.....	3
Table 2 - DPoE Interface and Reference Point Descriptions	6

Superseded

This page left blank intentionally.

Superseded

1 INTRODUCTION

Comcast Corporation, Time Warner Cable, and Bright House Networks collaborated to develop the interoperability requirements to support business services products using Ethernet Passive Optical Network (EPON) as an access technology.

DOCSIS Provisioning of EPON (DPoE) is a joint effort of operators, vendors, and suppliers to support EPON technology using existing DOCSIS-based back office systems and processes.

Ethernet PON or EPON is an [802.3] standard for a passive optical network (PON). A PON is a specific type of multi-access optical network. A multi-access optical network is an optical fiber based network technology that permits more than two network elements to transmit and receive on the same fiber. Appendix I in [DPoE-SP-ARCHv1.0] has a more detailed explanation of multi-access optical networks.

This version of the DPoE specifications is focused on DOCSIS-based provisioning and operations of Internet Protocol (IP) using DOCSIS High Speed Data (HSD), or IP(HSD) for short, and Metro Ethernet Forum (MEF) services. DPoE Networks offer IP(HSD) services functionally equivalent to DOCSIS networks, where the DPoE System acts like a DOCSIS CMTS and the DPoE System and DPoE Optical Network Unit (ONU) to appear to act like a DOCSIS CM.

1.1 DPoE Technology Introduction

DPoE technology was established with the following common requirements already developed by operators. Each of the participant operators had previously selected 1G-EPON and 10G-EPON as the appropriate technology for one or more applications. EPON is a widely-deployed technology with a sufficient and large supply of vendors offering a variety of products for each component of the access network. 10G-EPON technology is now becoming available and is backwards compatible with 1G-EPON. A 1G-EPON network can be incrementally upgraded to 10G-EPON, adding or replacing ONUs one at a time if required. 1G-EPON and 10G-EPON are compatible with [SCTE 174] (RFoG).

The EPON protocol [802.3ah] and the amendment for 10G-EPON [802.3av] support a centralized operator-based controller (OLT) architecture with low cost Layer 2 access devices (ONU). The basic service mapping architecture in EPON is to map Ethernet (or IP) frame header information (such as addresses, IP DiffServ Code Points, Ethernet Q tag, S-VLAN/C-VLAN ID, ISID, bridge address, or other marking) to a logical circuit called a Logical Link Identifier (LLID) in [802.3ah]. The service function is similar to that used in DOCSIS networks in many ways because it is based on a centralized scheduler and uses an LLID which functions like an SID, supports both unicast and broadcast, and has other similarities.

Existing [802.3ah] EPON systems do interoperate within the strict definitions of 1G-EPON. Experience with lab testing, field trials, and deployments has shown operators that 1G-EPON OLT and ONU systems typically only interoperate with a single port ONU. This is because [802.3ah] specifies the interfaces on the PON (the DPoE TU interface) but does not specify any of the other system interfaces. For example, an OLT from vendor A will register an ONU from vendor B, but it is not possible to construct a VLAN from the DPoE MN interface to an S interface. This is a well-recognized limitation of [802.3ah]. The challenge is that neither 1G-EPON nor 10G-EPON specify OAMP to forward traffic between NNI ports and the PON, or UNI ports and the PON. This is not different from other Ethernet standards. For example, if two Ethernet switches from two different vendors are connected, each switch must typically be configured independently. The challenge for EPON is that the remote device (the ONU) cannot be reached, and therefore cannot be configured. A solution to this problem must then be based on developing a common (standard) method of reaching the controller for the ONU, identifying the ONU capabilities, and providing that information to the OLT so that it can configure a working end to forwarding service (in both directions).

Superseded

Even if EPON had solved that provisioning challenge, there is no standard management interfaces for the ongoing operations and maintenance of the network, including fault management, performance management, security, etc. Operators already have fully working and scaled-out systems that solve these challenges for DOCSIS networks. One of the primary goals for DPoE specifications is to use the existing DOCSIS back office infrastructure to scale up EPON-based business services.

1.2 Scope

This specification identifies recommendations for the adaptation or additions to DOCSIS specifications that are required to support DOCSIS Provisioning of EPON (DPoE).

Security services may be divided into the following major areas:

- Subscriber data privacy, includes device authentication and key exchanges to verify that the device (and accompanying certificates) can insure data path encryption for subscriber data
- Subscriber network security, including mechanisms to protect the subscriber's network from malicious network attacks and to prevent subscribers from malicious attacks against other subscribers
- Service provider network security, protecting the service providers' network as well as protecting subscribers from each other
- Device software and configuration, including measures used to verify the integrity of the devices, software on them, and their configurations. Without device security, the other forms of security could be compromised.

In this document, the term "subscriber" is used synonymously with "customer."

1.3 Goals

Collectively, the operators started the DPoE specification development to accomplish the following objectives:

- Identify and document the common requirements for triple play services for business customers over EPON.
- Adapt DOCSIS-based back office provisioning and operations models to EPON. This is the core objective of DPoE specifications.
- Develop consensus on additional requirements above and beyond DOCSIS specifications to take advantage of the capabilities of EPON. These are focused in the area of Ethernet services and MEF integration.
- Continue to leverage the supply chain and economic benefits of a large base of suppliers and high-volume supply chain in optics, subsystems, and network systems based on a commodity EPON technology. Doing so requires adapting operator processes and networks to the EPON system rather than making any changes to the EPON systems.
- Positioning DPoE specifications to continue to leverage those same benefits for 10G-EPON.
- Work with the established EPON vendor community to assure that these strategies can be effective to mutually develop DPoE Networks, and to create a marketplace for success for multiple vendors to provide solutions for the variety of needs within the operator environment.

1.4 Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

Superseded

"MUST"	This word means that an item is an absolute requirement of this specification.
"MUST NOT"	This phrase means that the item is an absolute prohibition of this specification.
"SHOULD"	This word means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.
"SHOULD NOT"	This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
"MAY"	This word means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

1.5 Organization of Specifications

The DPoE specifications are organized around existing DOCSIS specifications. The purpose of matching DPoE specification documents to existing CableLabs DOCSIS, IEEE, IETF, and MEF requirements is to facilitate the mapping of services from existing DOCSIS infrastructure to DPoE infrastructure, and to provide an organization that will be easy to maintain as related (referenced) standards, recommendations, or specifications undergo independent changes.

There are two types of documents in the DPoE specifications. The first includes informative and requirements documents called specifications that detail the specific requirements for products claiming compliance with the specifications. The DPoE specifications also include a new kind of document that does not fit into any of the above categories. The IP Network Elements (IP NE) Requirements [DPoE-SP-IPNEv1.0] are a set of common requirements for the management of IP network elements that operators have developed, which are above and beyond the requirements in DOCSIS specifications, but are nonetheless required in DOCSIS CMTS products today. These are not specifications because no new protocols or algorithms are provided. Most of the requirements in IP NE are existing requirements based on IEEE, IETF, or other network management standards.

The DPoE documents are detailed in Section 1.6 of this document and duplicated, for reference, in each of the DPoE specifications.

1.6 DPoE Specifications

This document is one in a series of eight (8) documents comprising the DPoE specifications. Collectively these documents represent the operators' requirements for EPON-based commercial services.

Table 1 - DPoE Specifications

Document	Document Title	Description
DPoE-SP-ARCHv1.0	DPoE Architecture Specification	DOCSIS Provisioning of EPON introduction, architecture, and narrative. Specifies fundamental architectural requirements (those that apply to more than one specification). Explains the purpose of each document below.

Superseded

Document	Document Title	Description
DPoE-SP-OAMv1.0	DPoE OAM Extensions Specification	Extensions beyond [802.3ah] and [802.3av] requirements.
DPoE-SP-PHYv1.0	DPoE Physical Layer Specification	Using the EPON PHY, the DPoE PHY specification makes mandatory some options within EPON and adds some additional requirements.
DPoE-SP-SECv1.0	DPoE Security and Certificate Specification	Specifications for support for DOCSIS network and system interfaces to provide transparent support of DOCSIS device authentication, code verification, and additional security for a DPoE implementation.
DPoE-SP-IPNEv1.0	DPoE IP Network Element Requirements	Best practices and operator requirements for IP network element management and operations. This document includes CMTS-like IP router requirements. This document recommends practices not currently covered by any existing DOCSIS specifications.
DPoE-SP-MULPIv1.0	DPoE MAC and Upper Layer Protocols Requirements	Specifications for support of a subset of DOCSIS 3.0 MULPI functionality with additional EPON requirements.
DPoE-SP-MEFv1.0	DPoE Metro Ethernet Forum Specification	Specifications for Metro Ethernet services added to DOCSIS static configuration provisioning model.
DPoE-SP-OSSIV1.0	DPoE Operations and Support System Interface Specification	Specifications for support of a subset of DOCSIS 3.0 OSSI functionality with additional EPON requirements.

1.7 Reference Architecture

The DPoE reference architecture identifies the elements that a DPoE Network minimally requires to illustrate and communicate the physical hardware and logical software interfaces between the functional subsystems of the DPoE architecture. The principal elements in the architecture are the DPoE System that resides in the operator network, and the DPoE ONU which may be an off the shelf EPON ONU, EPON SFP-ONU, or an EPON ONU with additional subsystems. The remaining elements in the architecture are existing servers and systems in the operator's network. All of the server elements have connectivity through an IP (TCP/IP) network. Transport of bearer traffic, and (in some cases) Layer 2 OAM signaling are available through either IP or Layer 2 Ethernet-based Network Interfaces.

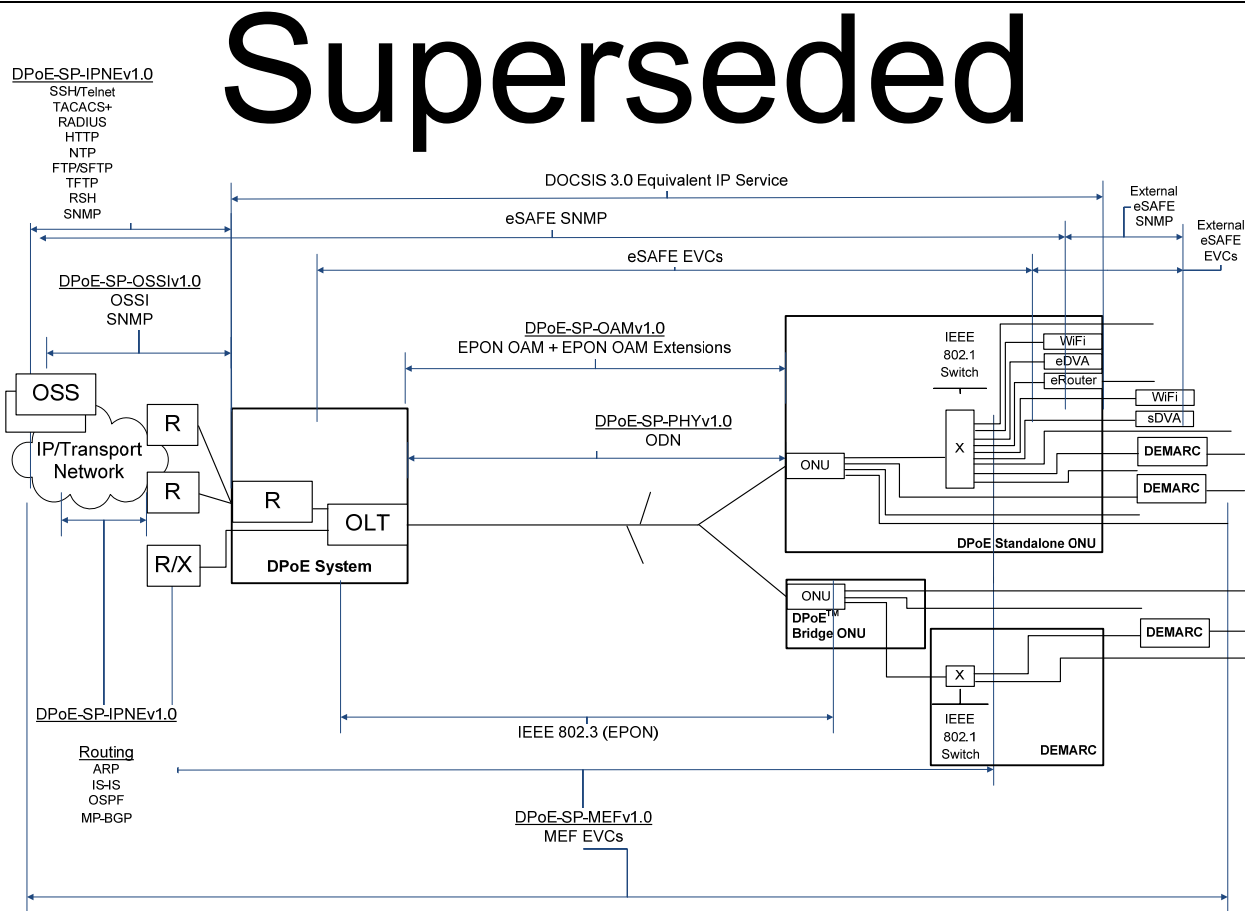


Figure 1 - DPoE Reference Architecture

1.8 DPoE Interfaces and Reference Points

The DPoE interfaces and reference points provide a basis for the description and enumeration of DPoE specifications for the DPoE architecture. Each interface or reference point indicates a point between separate sub-systems. The reference points have protocols that run across them, or have a common format of bearer traffic (with no signaling protocol). All of the interfaces are bi-directional interfaces that support two-way communications. The protocols in DPoE specifications operate within different layers based on the [802.3], [802.1], IETF, MEF, and CableLabs specifications. The C reference points are uni-directional for upstream (C_O) or downstream (C_S) classification, respectively.

Superseded

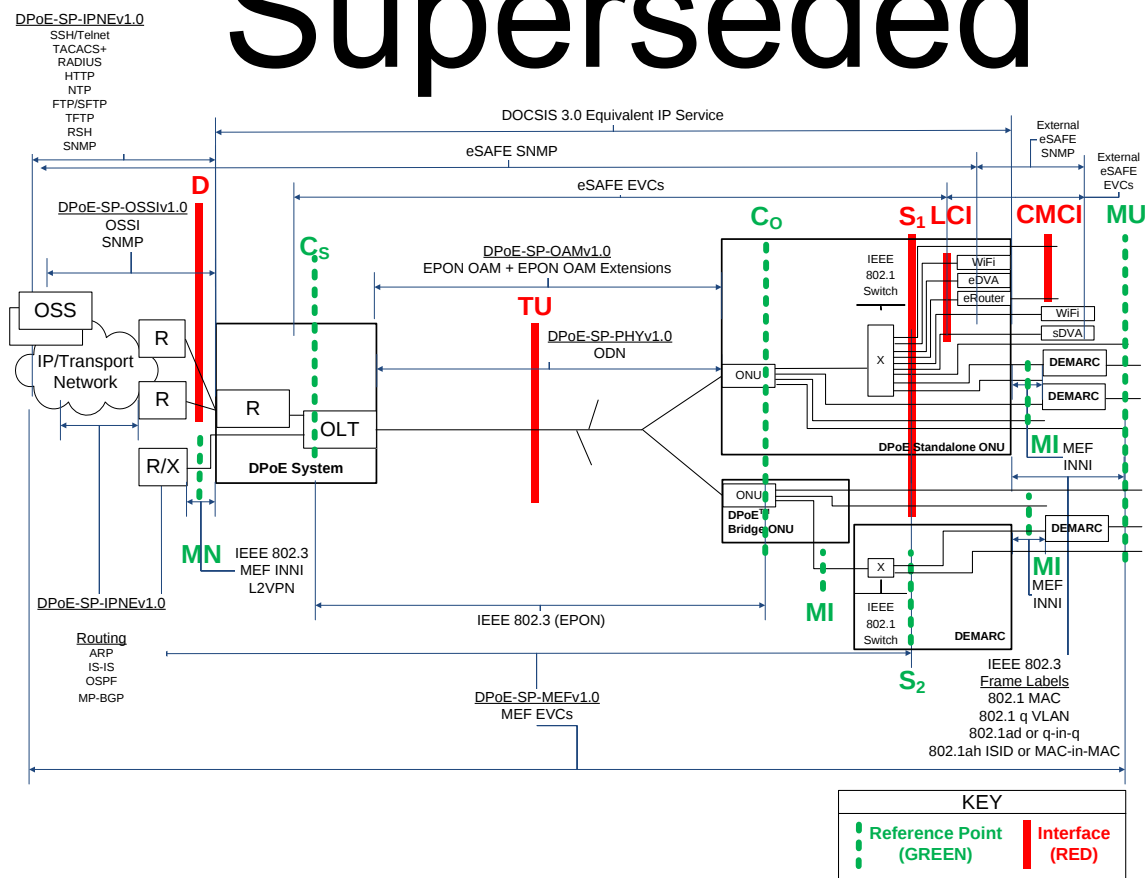


Figure 2 - DPoE Interfaces and Reference Points

Table 2 - DPoE Interface and Reference Point Descriptions

Interface or Reference Point		Interface or Reference Point Description
MN		The MN interface is an [802.3] interface for Ethernet (or MEF or L2VPN emulated) services only. It serves the role of a MEF INNI or L2VPN NSI. It is an NNI for Metro Ethernet services only.
D		The D interface is the DOCSIS IP NNI interface. It is an operator network facing interface, sometimes called a Network to Network Interface (NNI) or Network Systems Interface (NSI) in DOCSIS specifications. The D interface allows a DPoE System to communicate with an IP network. The D interface carries all IP management traffic including OSSI and IP NE traffic. The D interface carries all DOCSIS IP service traffic.
TU		The TU interface is a short form of expressing the interface between the DPoE System and the DPoE ONU.
C		The C reference point is used for explanation of traffic ingress to a DPoE classifier.
	C _O	The C _O reference point is used for explanation of traffic ingress to a DPoE ONU upstream classifier.
	C _S	The C _S reference point is used for explanation of traffic ingress to a DPoE System downstream classifier.

Superseded

Interface or Reference Point		Description
S		The S interface is an IEEE 802 interface. The S interface may be an internal interface (such as [802.3] across a GMII SERDES or XGMII interface in an SFP-ONU, SFP+ONU or XFP-ONU) or it may be an external Ethernet interface. S₁ is an interface for a DPoE Standalone ONU. S₂ is a reference point used for explanation of services with the DPoE Bridge ONU.
	S ₁	The S ₁ interfaces are the general case of all interfaces on a DPoE Standalone ONU. S ₁ interfaces may be CMCI, LCI, MI, or MU interfaces.
	S ₂	The S ₂ reference point is used for explanation of traffic ingress to and egress from interfaces on a DEMARC device in a DPoE System. Although there are no specifications or requirements for the S ₂ reference point, informative text refers to the S ₂ reference point to provide the full context for the use of a DPoE Bridge ONU in a DEMARC device providing Metro Ethernet services.
LCI		The Logical CPE Interface (LCI) interface is an eDOCSIS interface as defined in [eDOCSIS]. The eDOCSIS architecture is [802.1d] MAC based according to the DOCSIS 3.0 specifications; however, DOCSIS L2VPN clearly supports [802.1q] switching. In practice, therefore, the eDOCSIS interface consists of a DOCSIS classifier and [802.1] switch as illustrated. The function of a DOCSIS classifier is in part replaced by forwarding (tagging and encapsulation) in MEF and in part covered by classifiers in [DPoE-SP-MULPIv1.0].
CMCI		CMCI is the DPoE interface equivalent of the DOCSIS Cable Modem CPE Interface as defined in [CMCIv3.0]. This is the service interface for DOCSIS-based IP services.
MI		MI is usually an S interface (or S reference point) that operates as a MEF INNI. A DPoE ONU that provides a MEF INNI has an MI interface. A DPoE ONU can have MU as an interface and an MI reference point on different S interfaces in a single DPoE ONU. The MI interface or reference point is an [802.3] interface (or reference point) between a DPoE ONU and a DEMARC device.
MU		MU is usually an S interface (or S reference point) that operates as a MEF UNI. A DPoE ONU that directly provides a MEF UNI (MU) interface has MU as an interface. A DPoE ONU can have MU as an interface and an MI reference point on different S interfaces in a single DPoE ONU. The MU interface or reference point is an [802.3] interface (or reference point) between a DPoE ONU or a DEMARC device and a customer's equipment.

Superseded

2 REFERENCES

2.1 Normative References

In order to claim compliance with this specification, it is necessary to conform to the following standards and other works as indicated, in addition to the other requirements of this specification. Notwithstanding, intellectual property rights may be required to use or implement such normative references. At the time of publication, the editions indicated were valid. All references are subject to revision, and users of this document are encouraged to investigate the possibility of applying the most recent editions of the documents listed below. References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific. For a non-specific reference, the latest version applies.

[802.1d]	IEEE Std 802.1d™-2004, IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Bridges
[802.1q]	IEEE Std. 802.1q-2009, IEEE Standard for Local and Metropolitan Area Networks-Virtual Bridged Local Area Networks, January 2010.
[802.3av]	IEEE Std. 802.3av-2009, Amendment to IEEE Std. 802.3-2008: Physical Layer Specifications and Management Parameters for 10 Gb/s Passive Optical Networks.
[DPoE-SP-ARCHv1.0]	DPoE-SP-ARCHv1.0, DOCSIS Provisioning of EPON, DPoE Architecture Specification, Cable Television Laboratories, Inc.
[DPoE-SP-IPNEv1.0]	DPoE-SP-IPNEv1.0, DOCSIS Provisioning of EPON, IP Network Element Requirements, Cable Television Laboratories, Inc.
[DPoE-SP-MEFv1.0]	DPoE-SP-MEFv1.0, DOCSIS Provisioning of EPON, Metro Ethernet Forum Specification, Cable Television Laboratories, Inc.
[DPoE-SP-MULPIv1.0]	DPoE-SP-MULPIv1.0, DOCSIS Provisioning of EPON, MAC and Upper Layer Protocols Requirements, Cable Television Laboratories, Inc.
[DPoE-SP-OAMv1.0]	DPoE-SP-OAMv1.0, DOCSIS Provisioning of EPON, OAM Extensions Specification, Cable Television Laboratories, Inc.
[DPoE-SP-OSSiv1.0]	DPoE-SP-OSSiv1.0, DOCSIS Provisioning of EPON, Operations and Support System Interface Specification, Cable Television Laboratories, Inc.
[DPoE-SP-PHYv1.0]	DPoE-SP-PHYv1.0, DOCSIS Provisioning of EPON, Physical Layer Specification, Cable Television Laboratories, Inc.
[eDOCSIS]	Data-Over-Cable Service Interface Specifications, eDOCSIS Specification, CM-SP-eDOCSIS, Cable Television Laboratories, Inc.
[FIPS-140-2]	Federal Information Processing Standards Publication (FIPS PUB) 140-2, Security Requirements for Cryptographic Modules, June 2001.
[NIST 800-108]	NIST Special Publication 800-108, Recommendation for Key Derivation Using Pseudorandom Functions, October 2009.
[RFC 1750]	IETF RFC 1750, Randomness Recommendations for Security, December 1994.
[SECv3.0]	Data-Over-Cable Service Interface Specifications, Security Specification, CM-SP-SECv3.0, Cable Television Laboratories, Inc.

2.2 Informative References

Superseded

This specification uses the following informative references.

- | | |
|-------------|---|
| [802.1] | Refers to entire suite of IEEE 802.1 standards unless otherwise specified. |
| [802.1X] | IEEE 802.1X-2010, Port Based Network Access Control. |
| [802.3] | IEEE Std. 802.3-2008, IEEE Standard for Information technology-Telecommunications and information systems-Local and metropolitan area networks-Specific requirements, Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) access method and Physical Layer specifications. |
| [802.1ae] | "Media Access Control (MAC) Security", IEEE Standard Medium Access Control (MAC) Security, IEEE Std 802.1ae, June 2006. |
| [802.1ag] | IEEE Std 802.1ag™-2007, IEEE Standard for Local and metropolitan Area Networks – Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management, December 2007. |
| [802.3ah] | IEEE Std. 802.3ah-2004: Amendment to IEEE Std. 802.3-2003: Media Access Control Parameters, Physical Layers, and Management Parameters for Subscriber Access Networks, now part of [802.3]. |
| [CMCIv3.0] | Data-Over-Cable Service Interface Specifications, Cable Modem to Customer Premise Equipment Interface Specification, CM-SP-CMCIv3.0, Cable Television Laboratories, Inc. |
| [DOCSIS] | Refers to entire suite of DOCSIS 3.0 specifications unless otherwise specified. |
| [FIPS-197] | Federal Information Processing Standards Publication (FIPS PUB) 197, Advanced Encryption Standard, November, 2001. |
| [MULPIv3.0] | Data-Over-Cable Service Interface Specifications, MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.0, Cable Television Laboratories, Inc. |
| [OSSIV3.0] | Data-Over-Cable Service Interface Specifications, Operations Support System Interface Specification, CM-SP-OSSIV3.0, Cable Television Laboratories, Inc. |
| [RFC 3748] | IETF RFC 3748, Extensible Authentication Protocol (EAP), June 2004. |
| [RFC 5216] | IETF RFC 5216, The EAP-TLS Authentication Protocol, March 2008. |
| [SCTE 174] | SCTE 174 2010, Radio Frequency over Glass Fiber-to-the-Home Specification. |

2.3 Reference Acquisition

- Cable Television Laboratories, Inc., 858 Coal Creek Circle, Louisville, CO 80027; Phone +1-303-661-9100; Fax +1-303-661-9199; <http://www.cablelabs.com>
- FIPS Publications, NIST, 100 Bureau Drive, Gaithersburg, MD 20899-3460; Internet: <http://www.itl.nist.gov/fipspubs/>
- Internet Engineering Task Force (IETF) Secretariat, 48377 Fremont Blvd., Suite 117, Fremont, California 94538, USA, Phone: +1-510-492-4080, Fax: +1-510-492-4001, <http://www.ietf.org>
- Institute of Electrical and Electronics Engineers (IEEE), +1 800 422 4633 (USA and Canada); <http://www.ieee.org>
- SCTE, Society of Cable Telecommunications Engineers Inc., 140 Philips Road, Exton, PA 19341 Phone: +1-800-542-5040, Fax: +1-610-363-5898, Internet: <http://www.scte.org/>

Superseded

3 TERMS AND DEFINITIONS

3.1 DPoE Elements

- DPoE Network** This term means the entire network described in Figure 3 from the D or MN interface to the LCI, S, MI, or MU interface (see Figure 2 for interface and reference points), depending on the service being described. In no case does the term DPoE Network ever include a DEMARC device.
- DPoE System** This term means all of the collected elements that provide the DPoE function within the operator's network facilities. This includes the EPON OLT function, DOCSIS service functions required for the D interface, Metro Ethernet service functions required for the MN interface, and IP NE element management, routing and forwarding functions specified in [DPoE-SP-IPNEv1.0]. The DPoE System is depicted in Figure 3.
- DPoE ONU** This term means a DPoE-capable ONU that complies with all of the DPoE specifications. There are two types of DPoE ONUs. These are the DPoE Standalone ONU and the DPoE Bridge ONU.
- DPoE Standalone ONU** This term means a DPoE ONU that is a standalone ONU capable of providing IP or Ethernet services directly to customer premise equipment or transport of traffic to an external DEMARC device.
- DPoE Bridge ONU** This term means a DPoE ONU that is capable of [802.1] forwarding but cannot do all of the encapsulation functions required to be a DPoE Standalone ONU. Examples include an SFP-ONU and some simple EPON chipset-only based ONUs.
- DEMARC** Short form of "Demarcation Device." This term means the device, owned and operated by the operator that provides the demarcation (sometimes called the UNI interface) to the customer. Some architectures describe this device as the CPE (as in DOCSIS, DSL, or Broadband Forum Models) or the NID (as in the MEF model).

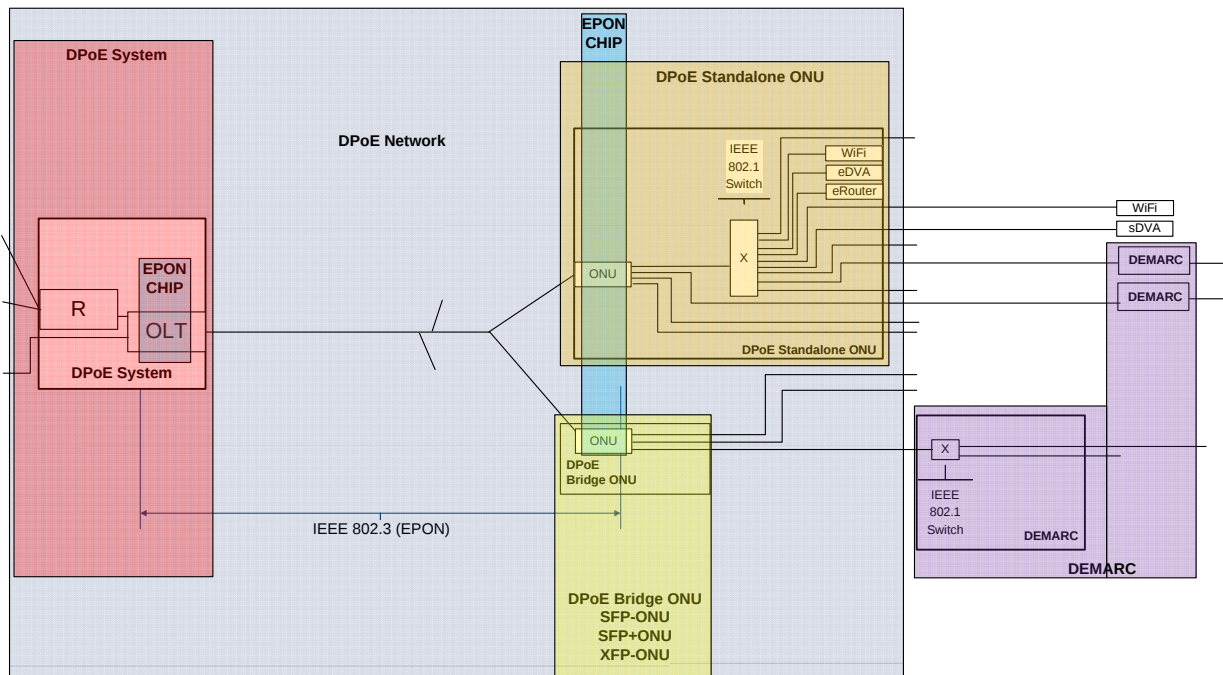


Figure 3 - DPoE Elements

Superseded

3.2 Other Terms

1G-EPON	EPON as defined in [802.3ah]
10G-EPON	EPON as defined in [802.3ah] and amended in [802.3av]
Cable Modem CPE Interface	CMCI as defined in [MULPIv3.0]
Customer Premise Equipment (CPE)	Customer Premise Equipment as defined in [DOCSIS]
Multi-Layer Switching (MLS)	A switch that can switch based on Layer 2, Layer 3, Layer 4, etc.
Ethernet Passive Optical Network (EPON)	Refers to both 1G-EPON and 10G-EPON collectively
EPON Operations and Maintenance Messaging (OAM)	EPON OAM messaging as defined in [802.3ah] and [DPoE-SP-OAMv1.0]; Ethernet OAM is not the same as EPON OAM; Ethernet OAM is [802.1ag]
Logical CPE Interface	LCI as defined in [eDOCSIS]
Network Interface Device (NID)	A DEMARC device in DPoE specifications

Superseded

4 ABBREVIATIONS AND ACRONYMS

This specification uses the following abbreviations:

AES	Advanced Encryption Standard
CMCI	Cable Modem CPE Interface
CPE	Customer Premise Equipment.
DA	Destination Address
DEMARC	Demarcation Device
DoS	Denial of Service
DPoE	DOCSIS Provisioning of EPON
EAP	Extensible Authentication Protocol
eCM	embedded Cable Modem
eDVA	embedded Digital Voice Adapter
EPL	Ethernet Private Line
EPON	Ethernet Passive Optical Network
EVC	Ethernet Virtual Connection
IP	Internet Protocol
LLID	Logical Link Identifier
MEF	Metro Ethernet Forum
MEN	Metro Ethernet Network
MI	MEF INNI Interface at a customer premise
MKA	MACSec Key Agreement protocol
MN	MEF INNI Interface to operators MEN
MPCP	Multi-Point Control Protocol
MPCPDU	MPCP Data Unit
MU	MEF UNI Interface
ODN	Optical Distribution Network
OLT	Optical Line Termination
OSC	Optical Splitter Combiner
ONU	Optical Network Unit
PHY	PHYsical Layer
PON	Passive optical network
QoS	Quality of Service
R	IP Router
SA	Source Address
SCB	Single Copy Broadcast

SFP	Small Form-factor Pluggable
SSD	Secure Software Download
UNI	User Network Interface
vCM	Virtual Cable Modem
X	IEEE Ethernet Switch (Generic)

Superseded

Superseded

5 OVERVIEW

DPoE security comprises functions within each of the DPoE Network elements and interfaces between the OSS, DPoE Network elements, and interfaces between the DPoE Network elements and subscriber devices or networks.

The security architecture is composed of discrete security functions and protocols that individually or collectively provide security for each of the security areas.

- Subscriber data privacy (TU interface)
- Subscriber network security (TU and C interfaces)
- Service provider network security (TU and C interfaces)
- Device software and configuration (D, TU, and C interfaces)

Security for the D interface is out of the scope of this specification. Some security functions for the TU interface rely on communication over the D interface.

Security on the TU interface includes subscriber data privacy, subscriber network security, service provider network security, and device software and configuration. Encryption is used on the TU interface to implement these features. Downstream encryption is always present; upstream encryption is optional, as the PON fiber upstream channel is generally considered physically secure. Provisioning and control of these services also requires configuration protocols that operate across the D interface. Those protocols are specified in the DPoE OSSI Specifications.

The S interface is the User to Network Interface for DPoE services. IP services rely on the use of a Layer 2 to Layer 3 address relationship (ARP in IPv4). IP services are limited to a single IP address and single MAC address at the CPE (across the S interface) for the IP (HSD) or DOCSIS equivalent service.

5.1 Subscriber Data Privacy

Subscriber data privacy includes device authentication and key exchanges required to verify that the device (and accompanying certificates) can ensure data path encryption for subscriber data. If a hostile device gets access to the EPON and is able to observe or collect the raw transmission on the fiber, the device will still not be able to recreate the data that is being communicated through DPoE system. In DOCSIS specifications, this is the traffic across CMCI or NSI interfaces. In DPoE specifications, this is the traffic across the TU interface.

As with DOCSIS specifications, the system and network from the D interface "northbound" or towards the core of the network (to the left in Figure 1) are assumed to be secure because those networks are physically separate from the subscriber access network and operate across complex multiservice transport networks that may have their own additional security. Transport, core IP, and Internet security are beyond the scope of this specification.

5.1.1 Unicast Traffic encryption

DOCSIS specifications support upstream and downstream unicast traffic encryption with the optional choice of DES or AES-128. The only cipher used in DPoE specifications is AES-128. In DOCSIS specifications, there is an option for message authentication with a message digest code, because some unencrypted control messages carry important system information and need to be authenticated on a message-by-message basis. DPoE specifications do not use this feature. Instead, all configuration and management messages are confined to an encrypted and authenticated channel between the authorized management source (the DPoE System) and the ONU, so that individual clear-text messages do not have to be individually authenticated.

Superseded

5.1.2 Unicast traffic key exchange algorithm

In current DOCSIS specifications, the BPKM protocol is used for key exchange between CMTS and CM. It uses private/public key cryptography to securely exchange keys. Public/private keys are taken from the X.509 certificate instead of being generated on the fly.

Depending on the cryptographic method configured for the PON, DPoE specifications use either DPoE OAM or the [802.1X] MKA protocol to exchange keys.

5.1.3 Multicast Traffic encryption

DOCSIS downstream multicast traffic encryption is outside the scope of this version of DPoE specifications, and so is not yet addressed in this document.

5.2 Service Provider Network Security

Service provider network security includes services to secure the provider's network against theft of service, denial of service, and services to protect subscribers from each other.

Secure provisioning plays a critical role in protecting DPoE ONUs and the DPoE Network against attacks, and in preventing theft of service. This section places requirements on the DPoE System and DPoE ONUs to support secure provisioning, including mandatory encryption of downstream data with optional encryption of upstream data; mandatory encryption of all control messages; security of software and configuration files, anti-MAC address spoofing, MAC address quantity limitation, DHCP/IGMP/BPDU suppression, anti-DoS attack mechanisms, etc.

5.2.1 Control Message Encryption

When encryption is enabled for a logical link, all control messages, for example DHCP, IGMP, ARP, MPCP, and OAM messages, MUST be encrypted by the DPoE System using the specified cipher suite, except those messages necessary for PON discovery and initial authentication used to establish this encryption.

5.2.2 IP Denial of Service attack mitigation

These are the measures that prevent malicious CPE from disrupting the system operation by presenting to the system stimulus that it cannot handle properly.

The DPoE System MUST provide a mechanism to prevent occurrence of the DoS (flooding) attack originating from a DPoE ONU. Examples include the ping of death, SYN flood, UDP flood, ICMP flood, TCP flood, etc. When the DPoE System discovers that the system is under any one of these attacks, the DPoE System MUST support an operator configurable option to drop IP packets if configured.

The DPoE System MUST support the capability to enable or disable each individual anti-DoS attack mechanism. The DPoE System MUST by default enable all anti-DoS attack mechanisms.

Requirements for IP network security across the D interface are out of the scope of this specification.

5.2.3 Ethernet Denial of Services Mitigation: Broadcast MAC forwarding

Malicious CPEs can send broadcast Ethernet packets across the network and disrupt normal communication.

Superseded

The Metro Ethernet service delivers a private Ethernet service and considers clients within the Metro Ethernet service to be trusted. In Metro Ethernet services, there is no need for filtering or blocking of MAC broadcasts.

The IP (HSD) service uses Ethernet as a transport from the DPoE System to the DPoE ONU and does not use Ethernet for any traffic between subscribers. The IP service will need to allow broadcast Ethernet traffic to be directed from the DPoE ONU to the DPoE System where the DPoE System will process the traffic. An example of a required broadcast is DHCP. DPoE Systems MAY offer operator configurable filters to control the data rate or block broadcast frames within a given VLAN. However, since the IP service does not carry frames between VLANs, this function is not required.

For a detailed description of Ethernet forwarding for IP (HSD), refer to [DPoE-SP-ARCHv1.0].

5.2.4 Limitation of the MAC address learning capacity

The DPoE ONU MUST support limitation of the number of MAC addresses learned on the S interface. The number of MAC address for each DPoE ONU S interface (port) MUST be configurable.

The DPoE ONU MUST support limitation of the MAC address learning capacity on per Ethernet port basis. The number of MAC addresses for each DPoE ONU port (across the C interface) MUST be configurable.

The DPoE ONU MUST NOT preserve nor remember learned MAC address when it is rebooted or power-cycled or re-registered in EPON. The DPoE ONU MUST store all learned MACs up to the configured maximum number of MAC addresses until a previously learned MAC address is aged out or the MAC cache is cleared.

The DPoE System MUST support clearing the MAC cache on a per DPoE ONU and per DPoE ONU S interface (or CMCI, LCI, MI, or MU) basis.

5.2.5 MAC address binding

The DPoE ONU MUST support the capability to configure specific SA MAC addresses and/or DA MAC addresses, which are bound to specific UNI ports, providing access from the specific devices or to specific devices, depending on the transmission direction. Any access for CPE devices whose source MAC addresses are not in the list of authorized addresses MUST be denied by the DPoE ONU.

5.3 eDOCSIS

eDOCSIS or eSAFE subsystems such as eDSG, eDVA, etc., are specified elsewhere. In cases where there is a fully operable embedded device within the DPoE ONU, those eSAFEs will need to implement the DOCSIS 3.0 SSD procedures from the OSS to the eSAFE device across the D and TU interfaces. Such SSD procedures are outside the scope of this document.

Superseded

6 ENCRYPTED FRAME FORMAT

6.1 1G Downstream-only Cipher Suite (1Down)

The DPoE System MUST only encrypt the DA through FCS bytes of the forwarded 1Down frame.

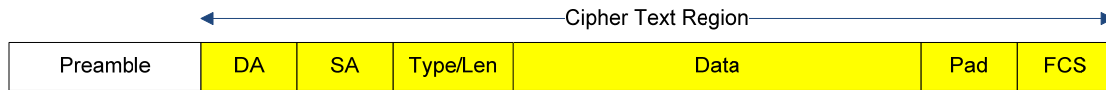


Figure 4 - Cipher Text Region (1Down)

The fifth octet of the preamble (immediately before the LLID) is used to carry security information.

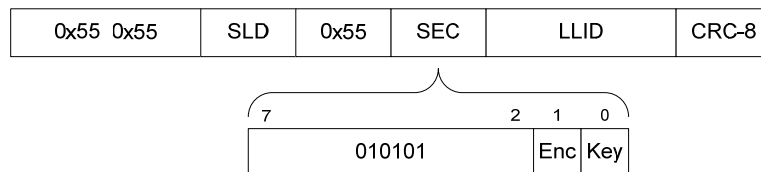


Figure 5 - Security Octet (1Down)

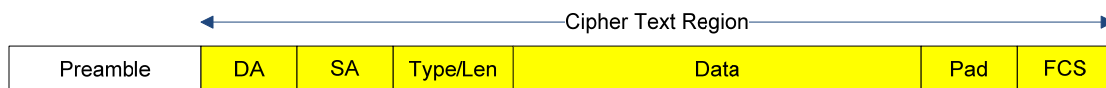
The DPoE System MUST set bit 1 in the security octet in the preamble of the 1Down frame to indicate that the data transferred is cipher text, and set bit 0 to the key identification number used to encrypt the frame. For 1Down encryption, bits [7:2] of the security octet are reserved, and the DPoE System MUST set them to 010101 (binary), matching the normal IDLE pattern. When encryption is disabled, the DPoE System MUST set the 1Down security octet to a value of 0x55.

6.2 10G Zero-Overhead Cipher Suite (10Down, 10Bi)

This section defines the frame format used with encryption on 10G EPON devices. Note that the upstream path uses this encryption method even with a 10G down / 1G up PON.

6.2.1 10G Zero-Overhead Frame Format

In the 10G mode, as in the 1G mode, the entire frame from DA through FCS is encrypted. Only the preamble remains as clear-text.



The fifth byte of the preamble (immediately before the EPON LLID) is the security octet.

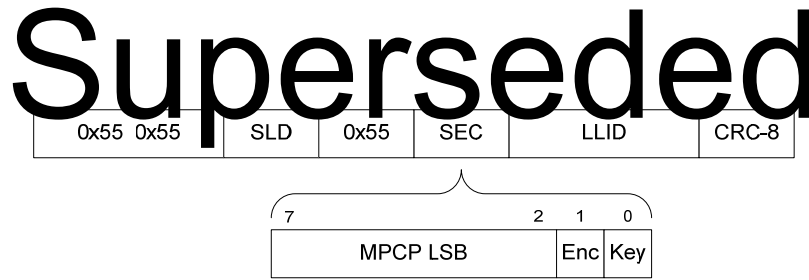


Figure 6 - Security Octet (10G Zero Overhead)

Fields in the security octet are:

Bits	Meaning
7:2	LSB of the MPCP time of the frame DA at the transmitter
1	1 if frame is encrypted; 0 if frame is not encrypted
0	Key identification number (0..1) of the key used to encrypt this frame

When encryption is disabled, the security octet has the value 0x55.

Superseded

7 KEY MANAGEMENT PROTOCOLS

7.1 1G Downstream-only Key Exchange Protocol

The key exchange protocol defined in this section **MUST** be used by DPoE System operating at 1 Gbps using the downstream-only encryption option. The key exchange protocol defined in this section **MUST** be used by DPoE ONU operating at 1 Gbps using the downstream-only encryption option.

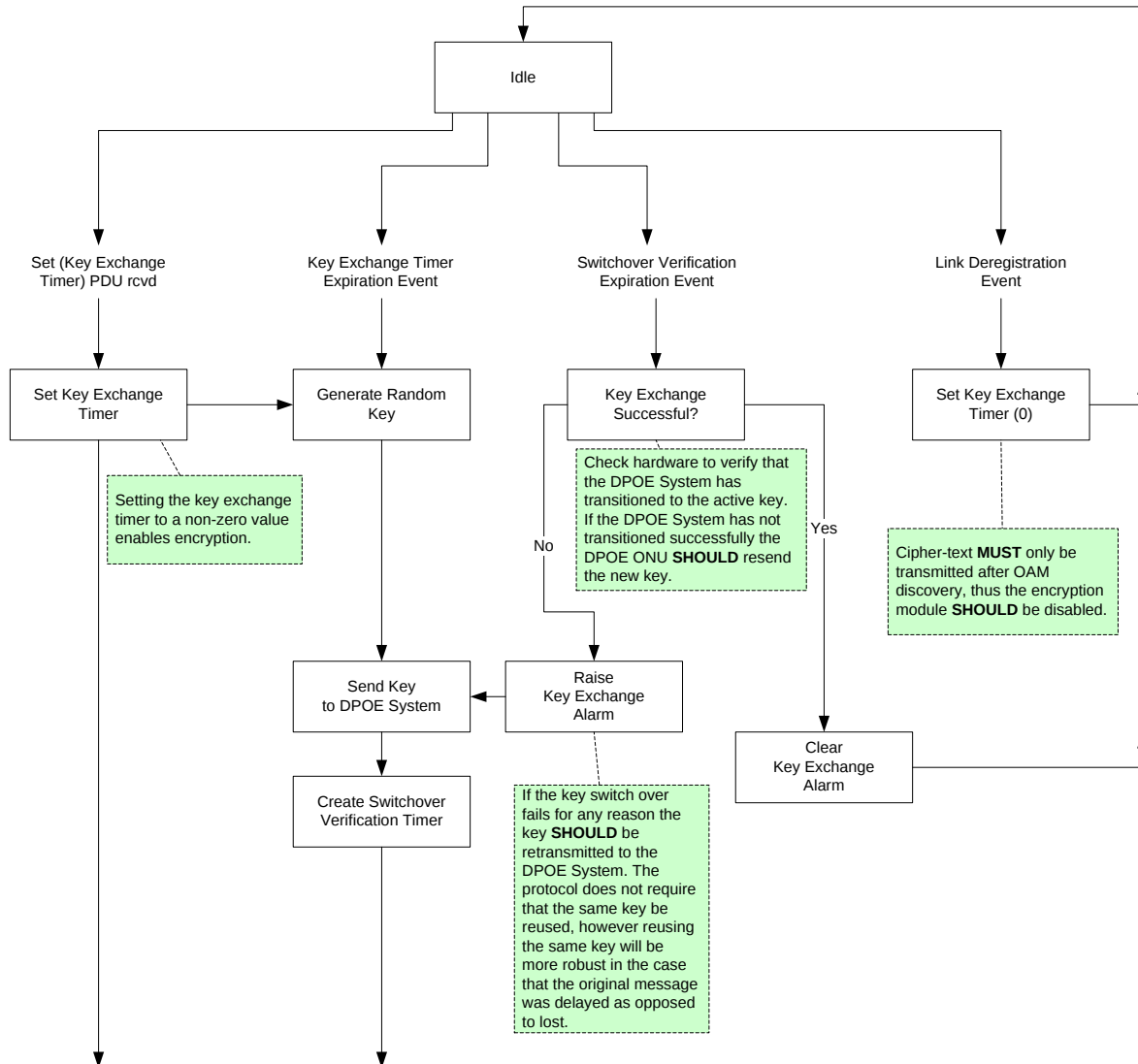


Figure 7 - DPoE ONU Key Exchange State Machine

The states in this state machine are described below.

7.1.1 Set Key Exchange Timer

Superseded

After MPCP and OAM discovery, the DPoE System configures a given logical link as a secure channel by sending a Set Key Exchange Timer OAM message to the DPoE ONU. This message contains a single timer value, Timeout, which will be used by the key exchange process. The DPoE ONU then transmits a Set Key Exchange Timer Ack indicating to the DPoE System that the link was successfully configured.

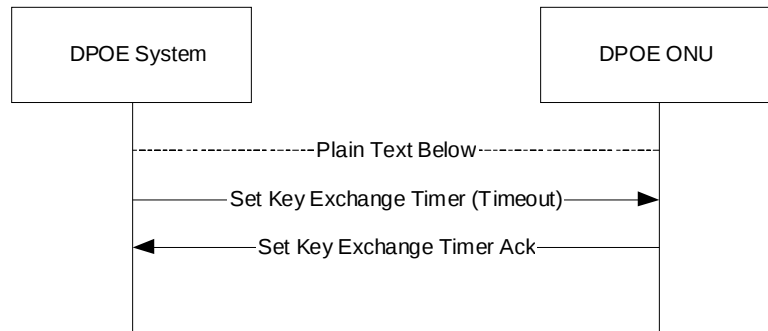


Figure 8 - Setting the Key Exchange Timer

The DPoE System SHOULD implement a retry mechanism in the event that the ACK is not received.

7.1.2 Create Key Exchange Timer

When a key exchange occurs, the DPoE ONU creates a timer initialized to the value specified by the Key Exchange Timer attribute that was used to enable encryption. The use of this timer is described in Section 7.1.7, Key Exchange Timer Expiration Event. The DPoE ONU MUST only accept a Key Exchange Timer value of at least 10 seconds. The DPoE ONU MUST NOT accept a Key Exchange Timer value of more than 65,535 seconds.

7.1.3 Generate Random Key

Once a given logical link has been configured as a secure channel, the DPoE ONU immediately generates a 128 -bit random key string associated with the link. The key string will be used by the DPoE System to encrypt data and by the DPoE ONU to decrypt cipher text received from the DPoE System.

7.1.4 Send Key to DPoE System

The DPoE ONU immediately transfers the key string along with a 1-bit key identification number to the DPoE System. The DPoE System uses the key string to generate cipher text using the cipher algorithm specified for the cryptographic method. (See Section 11, Cryptographic Methods.)

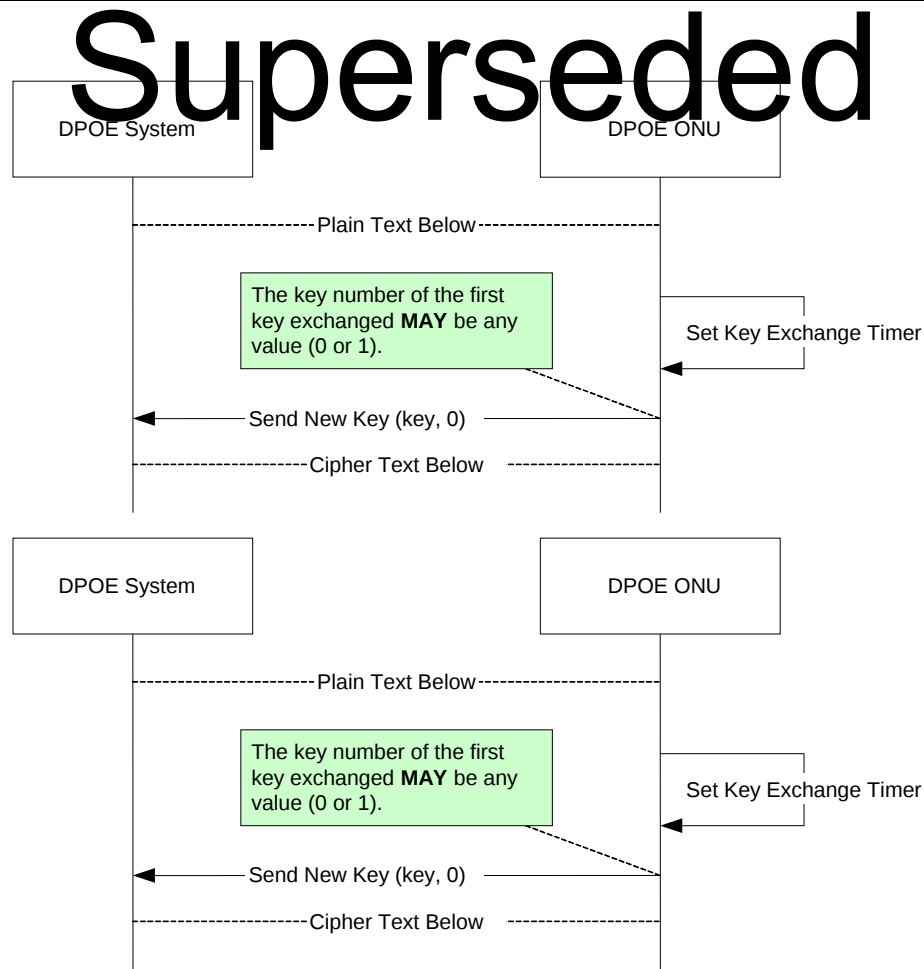


Figure 9 - Send Key to the DPoE System

7.1.5 Create Switchover Verification Timer (Optional)

After transferring the key to the DPoE System, the DPoE ONU SHOULD create a failsafe switchover verification timer. When this timer expires, the DPoE ONU verifies that the DPoE System is transferring cipher-text on the associated logical link using the latest key. If the link is still receiving plain-text, or data encrypted with the previous key, the DPoE ONU SHOULD retransmit the key to the OLT in the DPoE System. Refer to Section 7.1.7, Key Exchange Timer Expiration Event, for more information on Key Exchange.

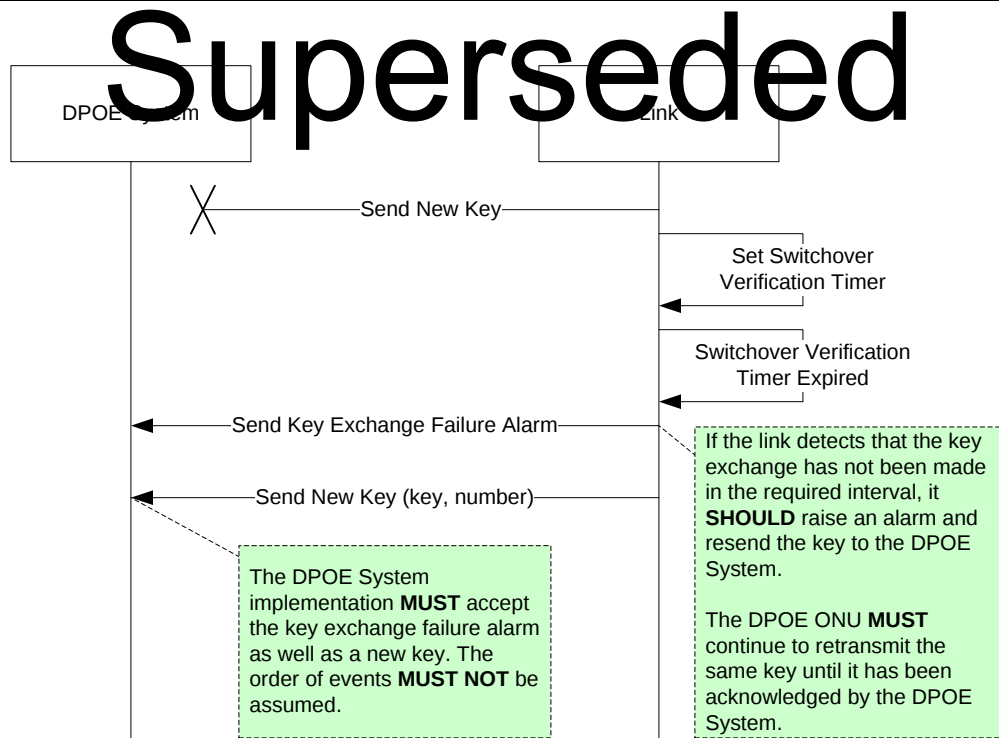


Figure 10 - Key Exchange Failure Detected by Link

The DPoE ONU **MUST** wait at least 1 second before verifying that the key exchange has taken place. This is long enough to guarantee that the DPoE ONU has received at least one encrypted MPCP GATE PDU from the DPoE System. The DPoE ONU **SHOULD NOT** wait more than $\frac{1}{2}$ of the actual key exchange timer value before verifying the key exchange.

7.1.6 Switchover Verification Event (Optional for ONU)

This event triggers the DPoE ONU to validate that the encryption key has been successfully updated by the DPoE System. If the key exchange has not occurred successfully, the DPoE ONU **SHOULD** send an appropriate alarm message to the DPoE System on the failing link and resend the key. Refer to Section 7.1.5, Create Switchover Verification Timer (Optional).

7.1.7 Key Exchange Timer Expiration Event

This event will trigger the DPoE ONU to generate a new security key and transfer it to the DPoE System. The DPoE ONU **MUST** retain the original key and transfer the new key with a different key identification number from the original. The DPoE ONU **MUST** accept and decrypt cipher-text with either the active or the next key. The key identification number transferred in the preamble of the cipher-text identifies the encryption key used by the DPoE System for that frame. The key exchange mechanism expires keys to limit the useful lifetime of a successful key extraction attack. Refer to Figure 11 below depicting two key exchange cycles.

On the DPoE System, expiration of the key exchange timer **SHOULD** generate an alarm to management software.

7.1.8 Link Deregistration Event

Link deregistration triggers the DPoE ONU to disable security for that link (only). Security should be re-enabled upon link discovery.

Superseded

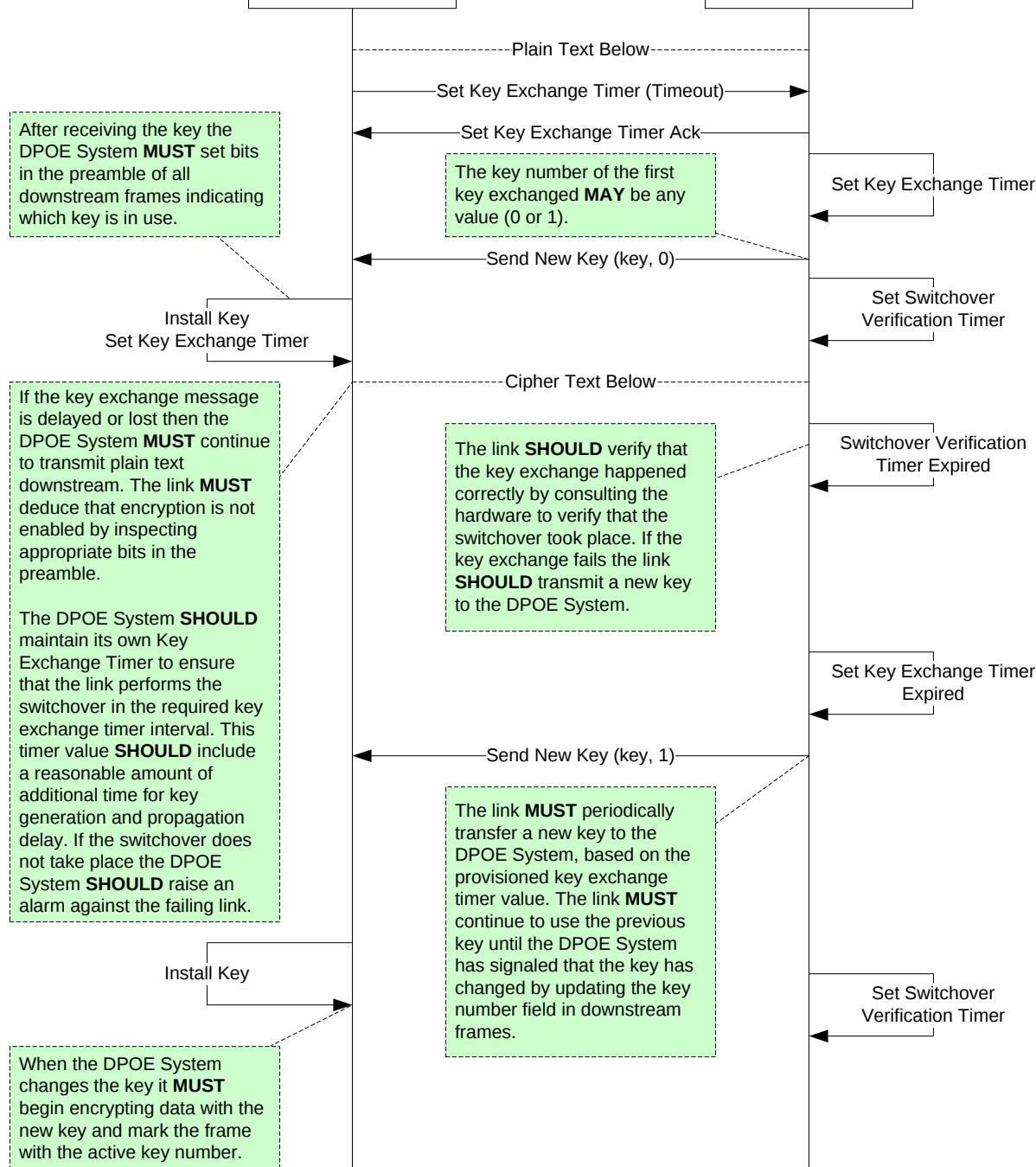


Figure 11 - Changing Keys On Timer Expiration

7.1.9 Detecting Key Exchange Failures at the DPoE System

The DPoE System SHOULD detect failure to receive a new key from the DPoE ONU within the provisioned key exchange time. The DPoE System MAY deregister the logical link when it fails to receive a new key from the DPoE

ONU. The DPoE ONU **SHOULD** retry the key exchange message (see Section 7.1.6, 8) whenever Verification Event (Optional for ONU), if more errors.

Superseded

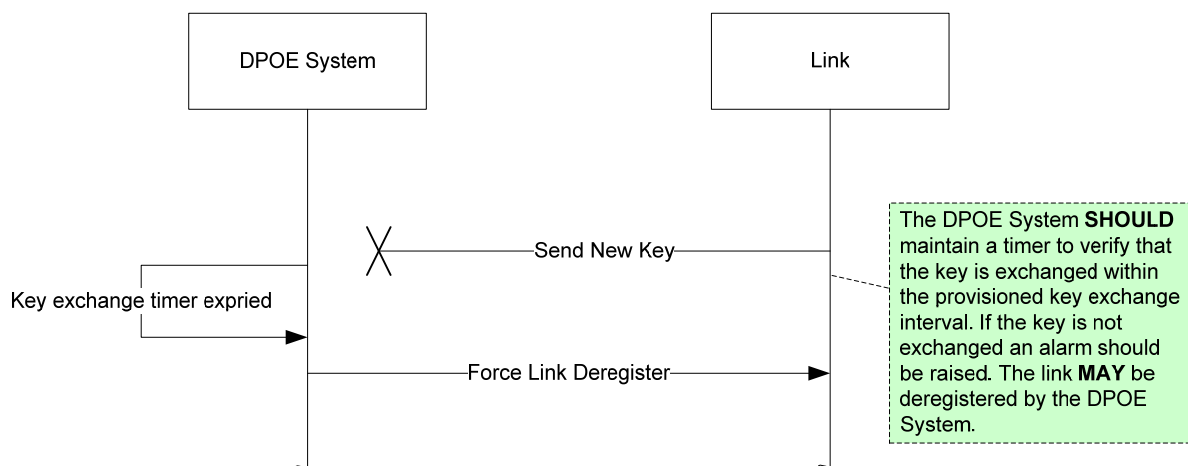


Figure 12 - Key Exchange Failure Detected By DPoE System

It is also possible that the DPoE System could fail to receive a response to the initial Set Key Exchange Timer sent to the DPoE ONU. The DPoE System **SHOULD** retry this message. If no response is received from the DPoE ONU, the DPoE System **SHOULD** deregister the logical link.

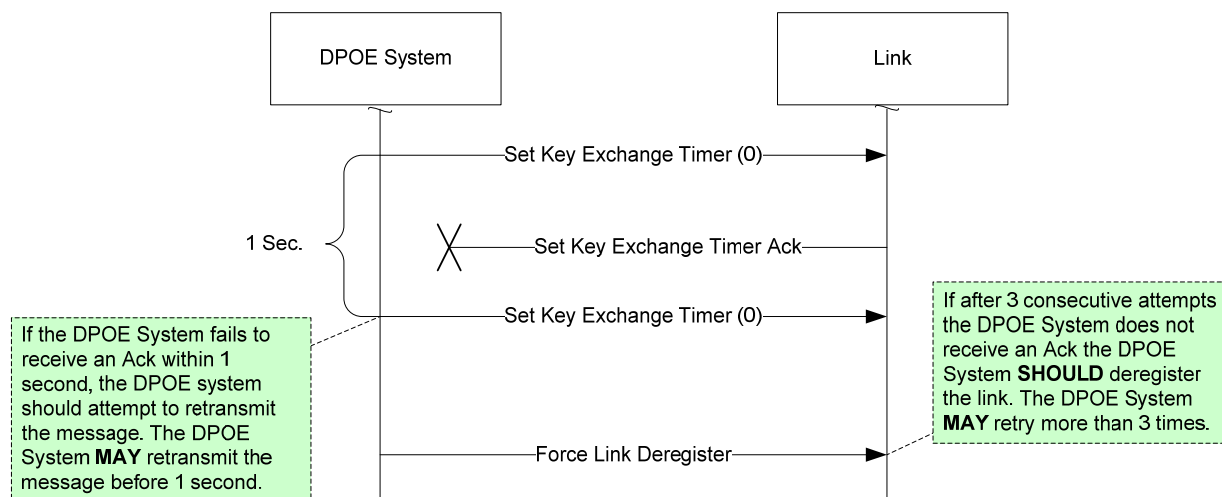


Figure 13 - Set Key Exchange Timer Failure Detected by DPoE System

7.1.10 DPoE OAM Key Exchange Messages

Key exchange message formats and attributes are defined by the [DPoE-SP-OAMv1.0] specification. The Key Exchange PDU type carries key values, while the Encrypt Key Expiry Time attribute controls the Key Exchange Timer interval.

7.1.11 Summary of 1Down Encryption Parameters

Superseded

Property	Description	Unit	Min	Max	Default
Key Exchange Timer	A repeating timer, an instance of which should be maintained by the DPoE ONU for every link configured as a secured channel.	N/A	N/A	N/A	N/A
Key Size	Number of octets of the key value to follow. Used to determine the length of the Encryption Key.	Octet	16	16	16
Key Security Key Encryption Key	String of bits used as input into the AES algorithm. A unique security key is generated by the DPoE ONU for each link configured as a secure channel.	N/A	0	$2^{128} - 1$	N/A
Encryption Key Expiry Time Key Exchange Timer Value	Time interval at which the DPoE ONU shall generate a new Key and transfer it to the DPoE System. A separate attribute is defined for each link associated with a particular DPoE ONU. Setting the attribute to 0 for a given link shall disable security on that link.	Seconds	1 (0)	60	0
Encryption Enabled	Flag present in every forwarded frame that is used to determine if the payload contains plain-text or cipher-text. 1= enabled, 0=disabled	N/A	0	1	0
Key Identification Number	The key identification number sent in every forwarded frame configured as a secure link and used by the DPoE ONU to determine which of the two possible keys associated with the link shall be used to decrypt the payload.	N/A	0	1	N/A

7.2 10G Downstream-only Key Exchange Protocol

The 10Down key exchange protocol is identical to the 1Down key exchange protocol. Keys are generated on the DPoE ONU and transmitted to the DPoE System in DPoE OAM Key Exchange PDUs.

7.3 10G Bidirectional Key Exchange Protocol

10Bi uses the MACSec Key Agreement (MKA) protocol defined in [802.1X].

Superseded

8 AUTHENTICATION AND ENCRYPTION

DPoE authentication and encryption is always "early" in the DOCSIS 3.0 sense, which is to say that it occurs before any DPoE ONU provisioning or user data traffic is allowed to be exchanged between the DPoE System and DPoE ONU.

A DPoE System **MUST NOT** configure a DPoE ONU or enable services either locally at the DPoE System, or at the DPoE ONU, until the authentication and encryption procedures configured for the DPoE Network have been completed.

8.1 DOCSIS and DPoE Authentication Comparison

Apart from terminology differences, packet formats, and details of the cipher suite, the authentication and key exchange process in DOCSIS specifications and the DPoE 10Bi method are very similar. Compare the two diagrams below, with the DOCSIS v3.0 method on the left and DPoE method on the right.

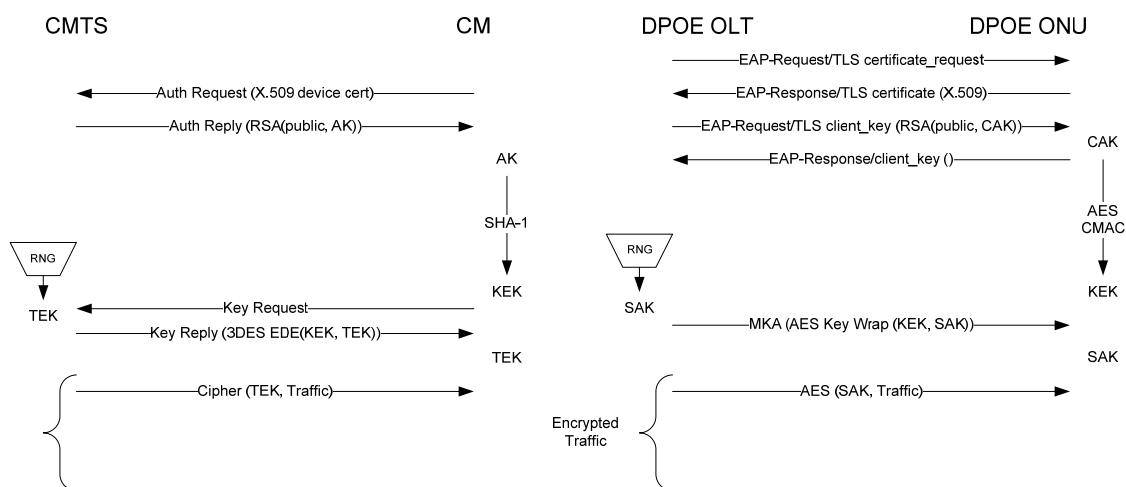


Figure 14 - Authentication in Bidirectional Methods

For downstream-only encryption modes, authentication remains nearly the same, while the key exchange differs in that the keys are generated at the ONU, transmitted upstream, and thus do not require encryption.

Superseded

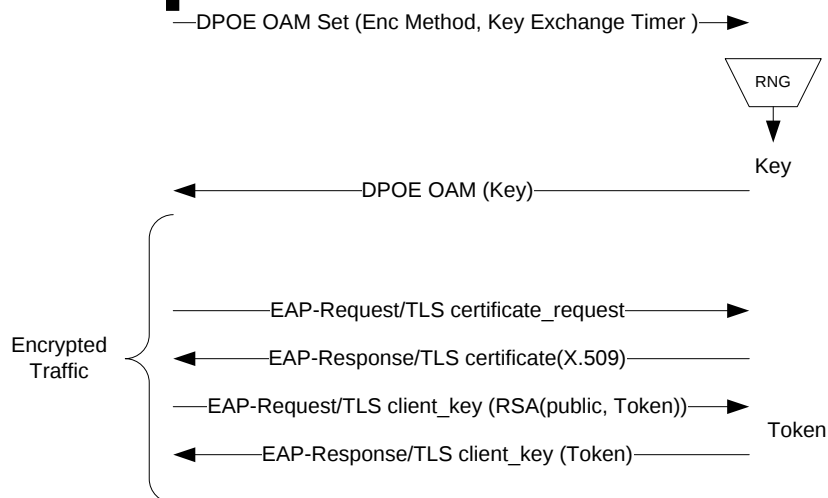


Figure 15 - Authentication in Downstream-Only Methods

Since the downstream-only mode does not use the key hierarchy anchored by public-key encryption of a session key during authentication, the downstream-only method **MUST** prove that the DPoE ONU holds the private key matching its certificate in some other way. To provide this proof, a random nonce of at least 128 bits is provided by the DPoE System in place of the CAK. This nonce is decrypted by the DPoE ONU using the private key and returned in the EAP-TLS client_key TLV. Note that this token is **NOT** a key, nor is it related to keys generated later. It is simply a random value to prevent a replay of the EAP Response.

When the bidirectional encryption mode is used, DPoE ONU **MUST NOT** return the CAK sent by the DPoE System, despite the similarity in protocol with the downstream-only mode.

8.2 DPoE ONU authentication

The DPoE Network uses device identity and authentication procedures functionally equivalent to DOCSIS. The DPoE Network uses existing DOCSIS protocols for all interfaces from the DPoE Network to the OSS for back office compatibility. However, the protocols and procedures for device authentication within the DPoE Network (from the DPoE System to the DPoE ONU across the TU interface) are different from those in DOCSIS. All of the TU interface protocols are distinct from the subscriber interfaces (C and S) and the OSS and NSI interfaces (D and M). Because none of these protocols are visible to the subscriber or the service provider, the DPoE specifications do not affect existing products, services, or operations for service providers. These specifications are for interoperability between DPoE Network elements (the DPoE System or DPoE ONU).

8.2.1 DPoE ONU MAC Address Identity

DOCSIS uses the DOCSIS CM MAC address as the identity of the CM. The identity is not implicitly trusted, but is the basic identify for all DOCSIS service OAMP. DPoE System **MUST** use the EPON ONU MAC address as the identity of the DPoE ONU.

When a DPoE ONU is powered on, each logical link reports its MAC address to the DPoE System through the MPCP discovery process as defined in [802.3ah] and [802.3av].

The DPoE System **MUST** support verification of the DPoE ONU's identity as authorized for the particular DPoE System port on which it is discovered. The DPoE System **MUST NOT** admit an authorized DPoE ONU on a

Superseded

different DPoE System port. The DPoE System MUST NOT admit an unauthorized DPoE ONU from accessing the network from any location. The DPoE System MAY use characteristics of the ONU (in addition to the MAC address and DPoE System port, such as round-trip time, to deny access to DPoE ONU that are unexpectedly relocated.

The first DPoE ONU to register with a particular MAC address and pass authentication MUST be the only DPoE ONU with that MAC address allowed by the DPoE System on the DPoE System port. Duplicate DPoE ONU MAC addresses MUST NOT be allowed by the DPoE System.

8.2.2 DPoE ONU Authentication

The DPoE System assumes that the DPoE ONU identity cannot be trusted until the DPoE ONU has been authenticated. Authentication of the device is a prerequisite for later software download, device configuration, service configuration, and service operation. The DPoE Network emulates the behavior of the DOCSIS system, although the implementation within the DPoE Network across the TU interface differs in terms of packet formats. To external systems across the D interface, the DPoE ONU device authentication (based on the MAC address and certificates) operates as specified [SECv3.0], [OSSIV3.0], and [DPoE-SP-OSSIV1.0].

The DPoE System MUST validate the DPoE ONU certificate using the procedures and criteria defined in [SECv3.0] and deny service to DPoE ONUs presenting invalid certificates.

8.3 Use of EAP-TLS for DPoE ONU Authentication

The packet format used to retrieve DPoE ONU certificates is EAP-TLS as defined in [RFC 5216]. This is a method of the EAP authentication framework, defined in [RFC 3748]. EAP is conveyed in Ethernet frames according to the EAP Over LAN (EAPOL) format as defined in [802.1X]. The summary of these specifications in this section is for informative purposes only. Normative requirements for use of these specifications are signaled below with the usual upper-case keywords.

A typical message sequence for EAP-TLS is shown in Figure 16.

Superseded

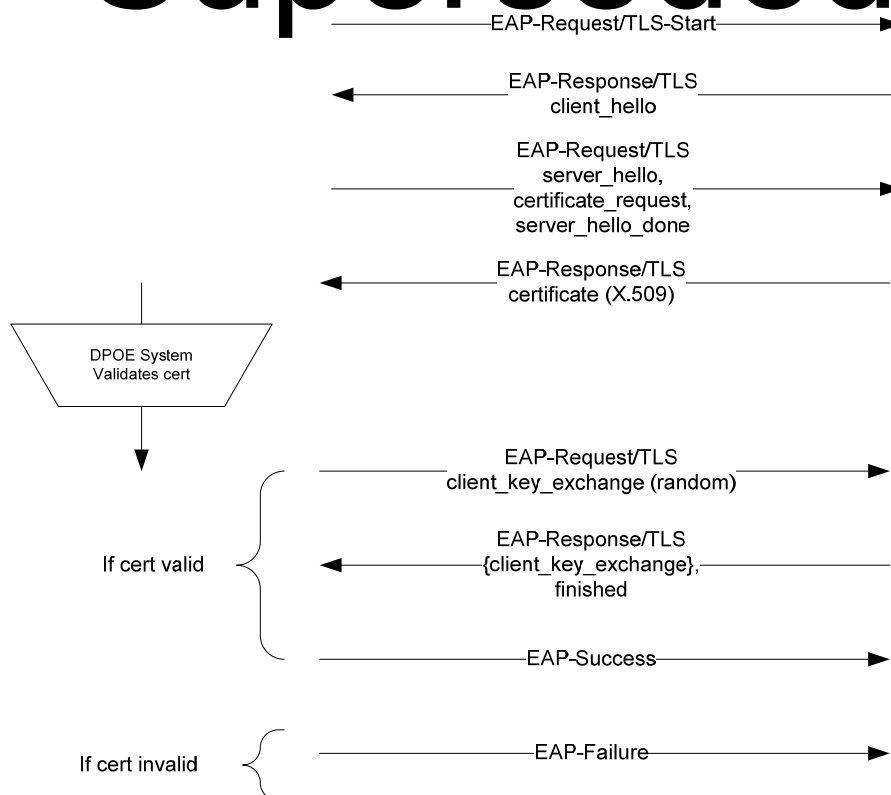


Figure 16 - EAP-TLS Message Sequence

The client_key_exchange TLV in the EAP Response from the DPoE ONU is present only for downstream-only operation, and contains proof that the DPoE ONU holds the private key matching the certificate sent earlier. In bidirectional modes, this TLV is omitted from the DPoE ONU response.

The EAP-TLS frame format is summarized in Figure 17.

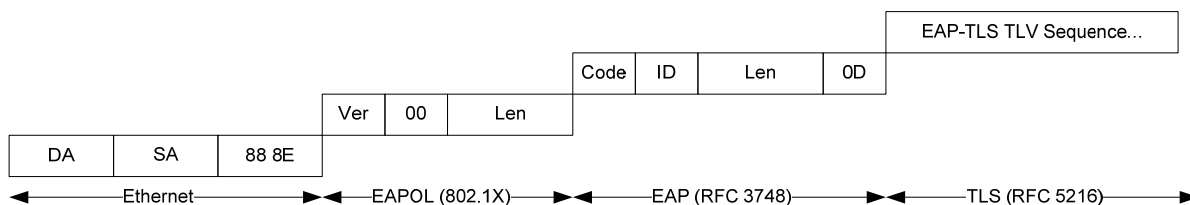


Figure 17 - EAP-TLS Frame Format

DA	Ethernet Destination Address
SA	Ethernet Source Address
88 8E	EtherType assigned to EAPOL
Ver	EAPOL Version
Type	EAPOL Type (0x00, "EAP")
Len	Length of EAPOL Packet Body to follow
Code	EAP Packet Type Code (1-4, Request / Response / Success / Failure)
ID	Arbitrary ID number that matches Responses to Requests

Len Length of packet to follow
 Type EAP authentication method (0=DPoE, 1=TLS)

Superseded

8.3.1 EAPOL Notes

The presence of a DPoE ONU is already well known to the DPoE System due to MPCP Registration and OAM Discovery, and so the Authenticator-Initiated option of EAPOL is used. A DPoE ONU MUST NOT send an EAPOL-Start frame to initiate Supplicant-Initiated Authentication.

Similarly, DPoE ONU deregistration is handled by MPCP. The DPoE ONU MUST NOT send an EAPOL-Logoff message before deregistering.

8.3.2 EAP Notes

The identity of a DPoE ONU is its MAC address. So, the DPoE System MUST NOT send an EAPOL-Request/Identity to the DPoE ONU to request its identity. Authentication begins with the DPoE System sending an EAP-Request/TLS-Start PDU to the DPoE ONU.

EAP-Success or Failure is determined by the DPoE System validation of the DPoE ONU certification as described in Section 13, X.509 Certificate Profile and Management. Note that a DPoE Network using bidirectional encryption may issue an EAP-Success frame to an DPoE ONU because the certificate is valid; yet that DPoE ONU may fail to decrypt the CAK if it does not possess the matching private key, and thus will deregister due to inability to decrypt MPCP Gate frames once encryption is enabled. In the downstream-only case, the DPoE ONU has proven its possession of the private key by returning a correct response to the DPoE System challenge. (See Section 13.2, Certificate Transport and Verification.) The DPoE System MUST send an EAP-Success only if the certificate is valid and the DPoE ONU challenge response is correct.

8.3.3 TLS Notes

TLS has the additional capability of negotiating encryption capabilities and modes, unfortunately not including the AES CTR mode used by DPoE Networks. Encryption capabilities are required by this specification and so are already known. Selection and enabling of encryption methods is carried out with DPoE OAM. The TLS encryption capability and negotiation TLVs are not used and MUST NOT be included in any TLS frame by a DPoE System. The TLS encryption capability and negotiation TLVs are not used and MUST NOT be included in any TLS frame by a DPoE ONU.

The length of time to authenticate an ONU SHOULD NOT exceed 300 seconds. (Most of this time is expected to be spent in RSA decryption of the session key or authentication token.)

Superseded

9 SECURE PROVISIONING

DPoE ONU configuration files are transferred from the OSS to the DPoE System over the D interface by the same mechanism as specified by [DOCSIS]. A DPoE System MUST implement the MIC and Extended MIC procedures for transferring and verifying DPoE ONU configuration files received from the OSS.

These configuration files are translated by the DPoE System into a series of DPoE OAM PDUs for transport across the TU interface to configure the DPoE ONU. The DPoE System MUST encrypt OAM PDUs for DPoE ONU configuration over the TU interface with the cryptographic methods defined in this specification.

9.1 ONU and CM Management Comparison

The DPoE System MUST NOT configure an IP address on a DPoE ONU that operates as the CM equivalent management address. In place of direct ONU configuration via an L3 configuration file transfer, the DPoE System operates a virtual CM (vCM) within the DPoE System. The DPoE System downloads the configuration for the "CM" into this vCM. Since the vCM resides within the DPoE System, which is always in the physical control of the operator, it (unlike a real CM) can be trusted.

The vCM controls the configuration of the DPoE ONU with DPoE OAM PDUs. The OAM messages are distinctly marked separate from subscriber traffic, and EPON OLTs and ONUs (in general) or DPoE Systems and DPoE ONU do not forward OAM messages across the D, C, or S interfaces. OAM messages only operate across the TU interface and are protected by encryption. Thus, there is no security threat based on a subscriber capability to create OAM messages and inject them into the control path.

This architecture provides additional security, and fast, simple, and low cost DPoE ONU implementation and operations. For example, if a DPoE ONU were configured by a private OLT and then later connected to a DPoE System in an operator network, it would be re-configured by the new operator OLT before any services were enabled. No matter what the state of the DPoE ONU configuration was prior to its registration, it will always be re-configured by OAM messages every time it re-registers with a DPoE System. In addition, the CVC process for the software (executable image) provides protection for firmware changes.

A DOCSIS CMTS performs a service authentication check during CM initialization. A CM may request the correct configuration file, but may then request services in the registration message that do not correspond to the configuration file contents. To prevent this error (or attack), DOCSIS recommends that the CMTS proxy TFTP file transfers to the CM, intercepting file transfers which happen to be configuration files destined for a CM, and comparing the contents of the registration message with the config file to be sure the services match.

In DPoE specifications, such a TFTP proxy is not required, since the DPoE ONU does not "request" service from the DPoE System over TFTP; service provisioning is always "pushed" from the DPoE System to the DPoE ONU using DPoE OAM.

Superseded

10 USING CRYPTOGRAPHIC KEYS

10.1 DPoE System

The DPoE System **MUST** be capable of maintaining two keys per logical link for encrypting downstream frames. The keys alternate to maintain continuous encryption of traffic when changing keys. The DPoE System **MUST** be capable of maintaining two keys per logical link for decrypting received upstream traffic. The DPoE System **MUST** be capable of decrypting received traffic with either of the keys, as indicated by the key identification number in each received frame. The DPoE System **MUST** also be capable of receiving unencrypted traffic even while encryption is enabled.

10.2 DPoE ONU

The DPoE ONU **MUST** maintain two keys per logical link for encrypting frames transmitted upstream. These keys alternate to maintain continuous encryption of traffic when changing keys. The DPoE ONU **MUST** maintain two keys per logical link for decrypting downstream frames. The DPoE ONU **MUST** be capable of decrypting received downstream traffic with either of the keys, as indicated by the key identification number in each received frame. The DPoE ONU **MUST** also be capable of receiving unencrypted traffic even while encryption is enabled.

10.3 Authentication of Dynamic Service Requests

Unlike DOCSIS, all DPoE OAM messages are encrypted. Individual messages are not signed with HMAC, and so the [802.1ae] CAK plays no role in signing control messages.

Superseded

11 CRYPTOGRAPHIC METHODS

Encryption is a mechanism used to guarantee security and privacy features. The physical properties of the PON medium used in the DPoE Network, as distinct from the RF over coaxial cable medium used in DOCSIS, affect the network design and requirements for privacy.

In a PON system, downstream frames are visible to all ONUs on a PON segment. Each frame is physically replicated by an optical splitter to every ONU. An ONU must permit only that data intended for that ONU to be forwarded. In EPON, this is accomplished by assigning each ONU a unique value known as a Logical Link Identifier (LLID). The OLT marks transmitted frames with the LLID, which is then used by the ONU to discard frames intended for other ONUs on the same PON segment. Unfortunately, the nature of a PON system leads to the possibility that a malicious ONU can potentially ignore the LLID and gain illicit access to downstream data intended for other ONUs.

Encryption is used in the downstream channel to ensure that a frame intended for one DPoE ONU can only be received by that ONU. Each ONU has a unique key, and frames intended for that DPoE ONU are encrypted with that key. Multicast groups can be created on the PON by the OLT sharing the key for an LLID with more than one ONU. The special multicast group known as the "broadcast" LLID is usually not encrypted, as all ONUs would have that key, and thus there would be no point in encrypting that data.

In the upstream direction, the properties of the optical splitter and commercially available optics modules make it effectively impossible for a DPoE ONU to successfully decode data transmitted by other users. The upstream wavelength is different from that used by the downstream receiver at the ONU. Even if the ONU receiver were modified, optical splitters typically have about a -55 dB return loss for reflection of the upstream signal, at least 26 dB more loss than the entire optical budget of the PON. Thus without tapping into the physical network on the trunk fiber upstream of all splitters, it is highly unlikely that a user can gain illicit access to upstream information.

Clearly a mechanism must be implemented to ensure that downstream data is securely transferred.

A DPoE System **MUST** implement the downstream encryption modes (1Down and 10Down) as described in this document. However, the implementation of additional upstream security mechanisms is not always required, as upstream data communication on the PON is naturally secure. DPoE System **MAY** implement the 10Bi method, including upstream encryption for use in cases where the upstream security of the PON may be suspect. A DPoE System and DPoE ONU **MAY** implement upstream encryption for a 1G bidirectional mode; such a method is outside the scope of this specification.

11.1 General encryption requirements

1. DPoE System data encryption **MUST** be supported on a per logical link basis. Some logical links may be encrypted on the PON while other logical links are not. This requirement retains compatibility with existing and future security standards and practices, and allows for the possibility of heterogeneous deployment, in which multiple security mechanisms are used on the same DPoE Network. Every downstream link **MUST** be configured at the DPoE System and DPoE ONU as an encrypted or non-encrypted channel. The DPoE System **MUST** be able to support both encrypted and unencrypted links on the same PON.
2. Each encrypted link **MUST** have a unique encryption key randomly generated by the DPoE System or DPoE ONU, as appropriate for the encryption method in use and type of logical link. If a non-deterministic random number generator is not available, the key generator **MUST** make use of sufficient entropy to generate a good quality seed as per [NIST 800-108], also following the guidelines in [RFC 1750].
3. DPoE System encryption **MUST** operate at full line rate of the DPoE Network.
4. ONU decryption **MUST** operate at full line rate of the DPoE Network.

Superseded

5. The DPoE System encryption process MUST NOT introduce jitter, though some additional constant latency is expected. The latency added by DPoE System encryption MUST NOT exceed 1 microsecond in either the upstream or downstream direction.
6. The ONU decryption process MUST NOT introduce jitter, though some additional constant latency is expected. The latency added by ONU decryption MUST NOT exceed 1 microsecond in either the upstream or downstream direction.

11.2 DPoE Cipher Suites

11.2.1 1G downstream-only Cipher Suite (1Down)

The DPoE System MUST implement the AES algorithm in Cipher Feed Back (CFB) Mode as described in this section when operating at 1 Gbps, and provide all necessary functions and interfaces to the software to support the key exchange described in Section 7.1.

The ONU MUST implement the AES algorithm in Cipher Feed Back (CFB) Mode as described in this section when operating at 1 Gbps, and provide all necessary functions and interfaces to the software to support the key exchange described in Section 7.1. 1Down uses AES 128 CFB mode to generate cipher-text, as described in Section 11.3. The key exchange protocol uses DPoE OAM, as defined in Section 7.1.

11.2.2 10G Downstream-only Cipher Suite (10Down)

DPoE Systems that support 10 Gbps downstream operation MUST support the 10Down encryption method as defined in this section.

10Down uses AES 128 bit CTR mode to generate cipher-text, as described in Section 11.4. The key exchange protocol is the same as for the 1Down method, as defined in Section 7.2.

11.2.3 10G Bidirectional Cipher Suite (10Bi)

DPoE Systems that support 10Gbps/10Gbps or 10Gbps/1Gbps MUST support the 10Bi encryption as defined in this section.

10Bi uses AES 128 bit CTR mode to generate cipher-text as described in Section 11.4. The key exchange protocol is the [802.1X] MKA protocol, as defined in Section 7.3.

11.3 1 Down Cryptographic Method

11.3.1 Encrypting Frames

The DPoE System MUST encrypt frames using AES in cipher-feedback mode (CFB).

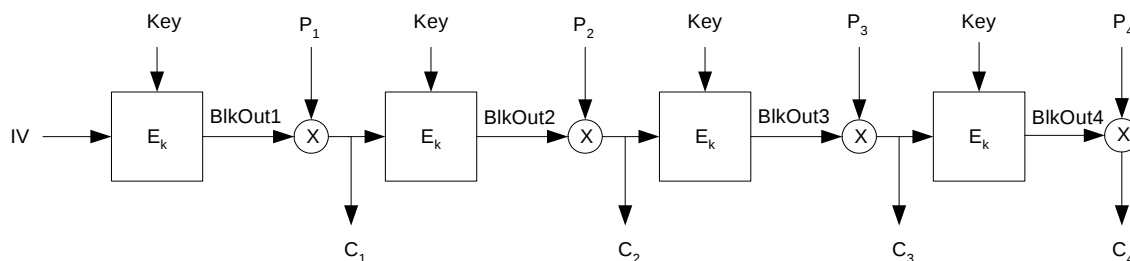


Figure 18 - Encrypting Data with Cipher Feedback Algorithm

Superseded

The AES algorithm is used to generate a cipher based on the input encryption key and a seed value known as the "Initial Vector" (IV). The IV must be known to both the host generating the cipher text and the host that must decrypt the text. (The IV may also be known to other hosts without affecting the security of the cipher.) The derivation of the IV for the 1Down encryption method is described in Section 11.3.3.

The cipher is logically XORed with the plain text P1 to produce the first 16 octet block of cipher text C1. This block of cipher text is then used as the IV for the next block of plain text (hence the term "cipher feedback"). This process is repeated until the entire frame is encrypted.

11.3.1.1 Handling Frames of Non-Ordinal Length

If the clear-text length (the entire MAC frame, DA through FCS) in octets cannot be divided by 16 without remainder, the final block of cipher text will be generated by first zero- extending the clear-text to a 16 octet boundary, encrypting a full 16 octet block, and then discarding the unneeded cipher-text that corresponds to the padding of the clear-text.

11.3.2 Decrypting Frames

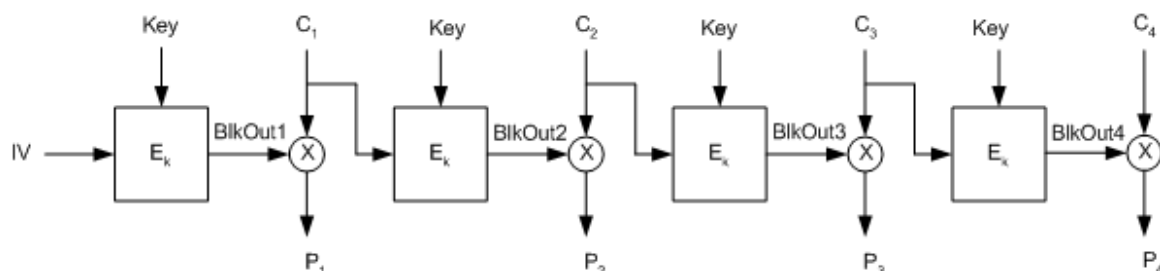


Figure 19 - Decrypting Data with Cipher Feedback Algorithm

To decrypt the frame, the IVs and key are used to create the original cipher blocks used to encrypt the data. XORing the cipher block with the cipher text produces the original plain text. Refer to Figure 19, the figure above which illustrates the decryption process for 4 blocks of cipher text.

11.3.3 Initial Vector (IV) Source

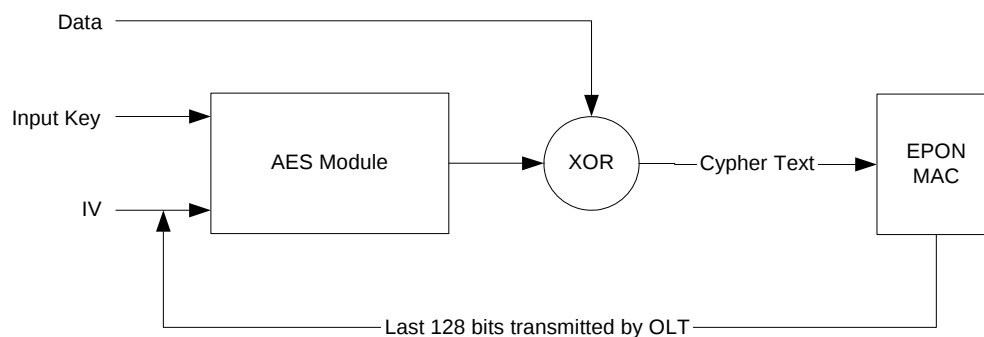


Figure 20 - Determining the Initial Vector

The IV is taken from the last 128 bits of the previous frame received from the EPON at the RS sublayer, which includes all downstream frames destined for any ONU regardless of LLID. Since the last 128 bits of the previous frame contains the FCS, the IV computation is a function of every bit of the previous frame. The initial vector is based on the last encrypted frame forwarded downstream on the EPON to any logical link, not the last frame forwarded downstream to any particular link.

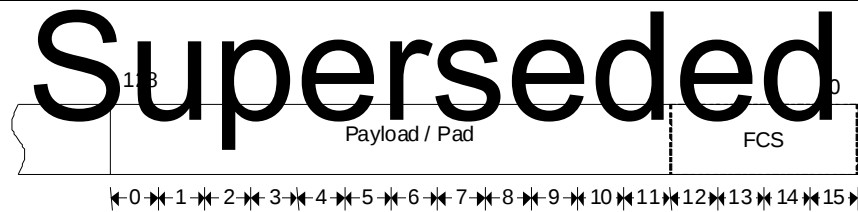


Figure 21 - Octet order within the Initial Vector (1G)

The octets in the initial vector are ordered starting with the first octet transferred (or received) and continuing sequentially in network byte order. The last four octets of the IV are those corresponding to the FCS of the previous frame.

11.4 10 Down / 10BiCryptographic Method

11.4.1 Encrypting Frames

The transmitter of frames in 10Bi mode MUST encrypt frames using the AES Counter (CTR) mode.

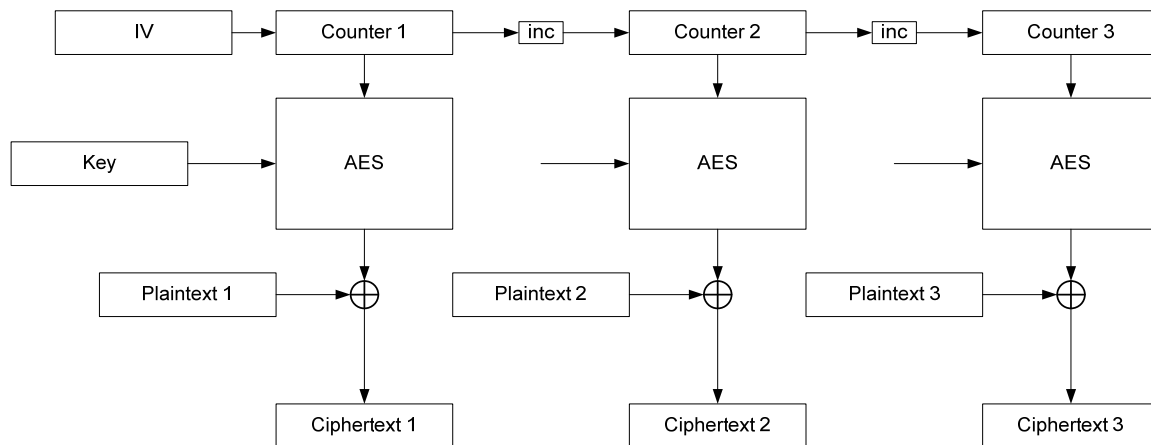


Figure 22 - Encrypting Data with CTR Mode

The AES algorithm is used to generate a cipher based on the input encryption key and a seed value known as the IIV. The IV must be known to both the transmitter generating the cipher text and the receiver that must decrypt the text. (The IV may also be known to other hosts without affecting the security of the cipher.) The derivation of the IV for the 10G cipher is described in Section 11.4.3.

In the second stage, the IV is used as input to AES along with the key. The AES output is logically XORed with the plain text, Plaintext 1, to produce the first 16 octet block of cipher text, Ciphertext 1. The counter is then incremented to produce the counter for the next input block, Counter 2. This process is repeated block by block until the entire frame is encrypted, as shown in Figure 22.

If the plain-text length in octets cannot be divided by 16 without remainder, the final block of cipher text will be generated by first zero-padding the plain-text to a 16 octet boundary, encrypting the data, and then discarding the excess cipher-text corresponding to the pad value.

11.4.2 Decrypting Frames

Superseded

To decrypt the frame, the IVs and key are used to create the original cipher blocks used to encrypt the data. XORing the cipher block with the cipher text produces the original plain text. The process is as shown in Figure 22, except with ciphertext as input and plaintext as the result of the XOR.

11.4.3 Initial Vector (IV) Source

In this method, the IV is formed by concatenating the transmitter's Source Address + the LLID + the MPCP time + a counter (called "Yincr") which increments for each 128-bit block from the start of the frame.

The transmitter Source Address is not the SA in each received frame, but is the Source Address of the peer of the link. That is, in the downstream, this is the SA of the OLT, not of the originator of the frame; in the upstream, this is the SA of the ONU for that logical link. These SAs have previously been learned by the encryption peers during the MPCP registration and OAM discovery phases. The 15-bit LLID value is extended to 16 bits with a leading 0 in the most significant bit (MSB) when forming the IV.

The MPCP time for the IV is the time of the first byte of the DA of the frame.

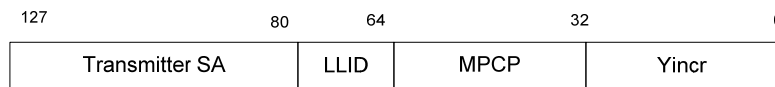


Figure 23 - Octet order within the Initial Vector (10G)

Since the MPCP time between OLT and ONU is allowed to vary to accommodate clock jitter, the MPCP receive time at the ONU will not necessarily exactly match the MPCP time when the frame was transmitted. This jitter is accommodated by carrying the 6 low-order bits of the MPCP clock from the transmitter to the receiver in the security octet of the preamble of the frame. The receiver then compares these bits to its own MPCP time to determine the precise value of the transmitter's MPCP clock when the frame was sent. The full 32-bit MPCP transmit time is corrected for jitter and used in the IV for the frame.

11.4.4 MPCP Jitter Correction

In the downstream direction, the ONU MPCP clock lags behind the OLT MPCP clock by the propagation delay between the two. This delay is already accounted for in the MPCP clock values, and does not require adjustment. However, the MPCP time at the receiver must be adjusted for jitter before it can exactly match the MPCP time used by the transmitter.

In the upstream direction, the OLT receive MPCP clock will be in advance of the ONU transmit MPCP clock value by the round-trip time (RTT) between the OLT and ONU, as discovered during MPCP registration, plus or minus jitter as allowed by [802.3ah]. In the upstream direction, the OLT subtracts the RTT from the receive time. The OLT must also correct the MPCP receive time for jitter as in the downstream direction.

To determine the MPCP time used by the transmitter to encrypt the frame, the MPCP time of a frame at reception is rounded up or down to the nearest MPCP value with the correct low-order bits according to the following procedure (or equivalent behavior). Recall from Section 6.2.1 that the least significant six bits of the MPCP time are carried in the security octet in the preamble of each frame.

```
lsb_MPCP : lowest 6-bit MPCP value sent in security octet
local_MPCP : local MPCP time of the 1st byte of the DA of the frame
corr_MPCP : high-order bits of MPCP value corrected for jitter
MPCP : final MPCP value to be used in IV for the frame
```

```
if lsb_MPCP[5] xor local_MPCP[5]
    // MPCP rollover has occurred at one end but not the other
    if (local_MPCP[4] == 1)
        // increment the local MPCP to the nearest future value which
```

Superseded

```

    // has the local_MPCP[5] = lsb_MPCP[5]
    corr_MPCP[31:5] = local_MPCP[31:5] - 1;
else
    // decrement the local MPCP to the nearest past value which
    // has the local_MPCP[5] = lsb_MPCP[5]
    corr_MPCP[31:5] = local_MPCP[31:5] - 1;
end
else
    corr_MPCP[31:5] = local_MPCP[31:5];
end
MPCP[31:0] = {corr_MPCP[31:6], lsb_MPCP[5:0]};

```

Yincr begins at a value of 1 at the DA of each frame, and is incremented with each successive 128-bit block of data in the frame.

Superseded

12 PHYSICAL PROTECTION OF SECURITY DATA IN THE ONU

DPoE ONUs MUST implement the requirements of [SECv3.0] for secure non-volatile storage of certificate and the corresponding public/private keys.

The DPoE ONU MUST store the certificate private key in a manner that deters unauthorized disclosure and modification. Also, a DPoE ONU SHOULD prevent debugger tools from reading the DPoE ONU certificate private key in production devices by restricting or blocking physical access to memory containing this key.

DPoE ONUs SHOULD use one-time-programmable (OTP) memory to store certificates and keys to make it more difficult to "clone" a valid DPoE ONU.

The DPoE ONU MUST meet [FIPS-140-2] Security requirements for all instances of private and public permanent key storage.

The DPoE ONU MUST meet [FIPS-140-2] Security Level 1, which requires minimal physical protection through the use of production-grade enclosures. The reader should refer to the cited document for the formal requirements; however, as a summary, the specifications require DPoE ONU chips to be of production-grade quality, including standard passivation sealing. The circuitry within the DPoE ONU MUST be implemented as a production-grade multi-chip embodiment as with an IC printed circuit board. The DPoE ONU MUST be contained within a metal or hard plastic enclosure, which may contain doors or removable covers.

Superseded

13 X.509 CERTIFICATE PROFILE AND MANAGEMENT

13.1 X.509 Certificate Verification

An X.509 certificate is used to authenticate DPoE ONU identity in a manner similar to DOCSIS cable modems. If a DPoE ONU duplicates (copies) another DPoE ONU MAC Address and presents it to the DPoE system during authentication, the DPoE system MUST prevent EPON registration of such a duplicate ONU. Each DPoE ONU is assigned a unique X.509 DPoE ONU certificate, which is programmed into it during the manufacturing process. Such a certificate contains information unique to the particular DPoE ONU, including the MAC address, manufacturer's serial number, and public key. The format of this certificate is as defined in [SECv3.0], Appendix III.

The certificates are signed by a chain of trust established through an intermediate CA and Root CA as defined in the appendix. Usually the signing process involves three certificates in a chain. The ONU certificate is signed by the intermediate CA, and the intermediate CA's certificate is signed by a Root CA.

13.2 Certificate Transport and Verification

In DOCSIS, the X.509 certificate is transported to the CMTS using the BPKM protocol. In DPoE Networks, the certificate is transported using EAP-TLS, as defined in [RFC 5216], over the EAPOL framework defined in [802.1X].

EAP transports authentication messages at the link layer without fragmentation. Therefore, each X.509 certificate MUST NOT exceed 1200 bytes in length, so that it will fit into a single EAPOL packet.

An EAP-Request PDU is sent by the authenticator (DPoE System) to retrieve the DPoE ONU certificate. Once the certificate is delivered, the DPoE System can verify the certificate. The DPoE System MUST validate the DPoE ONU certificate using the procedures and criteria defined in [SECv3.0] (considering the different DPoE root CA) and deny service to DPoE ONUs presenting invalid certificates. The DPoE System MUST deregister any ONU that fails authentication.

Once the certificate is validated, the DPoE System MUST further prove that the DPoE ONU that sent the certificate is also in possession of the private key that matches that certificate.

For DPoE Networks that use bi-directional encryption, proof of possession of the private key is provided as follows. The DPoE System sends an EAP-TLSPDU with a `client_key_exchange` TLV, which contains a randomly generated string of data encrypted with the DPoE ONU public key.

For DPoE Networks that use bi-directional encryption methods, this random string is used as the [802.1X] Connectivity Association Key (CAK) for the CA containing the OLT and DPoE ONU. Without the correct private key, the CAK cannot be decrypted by the DPoE ONU. Thus, derived traffic keys (SAKs) cannot be decrypted, and once encryption is enabled, the DPoE ONU will be unable to decrypt further traffic on the PON and will deregister. Only a DPoE ONU in possession of the correct public key will be able to remain registered and receive service. The `client_key_exchange` TLV of the Response by the DPoE ONU is empty. See Figure 14 and Figure 16.

For DPoE Networks that use the downstream-only encryption methods, proof of the private key is provided by the DPoE ONU decrypting a nonce randomly generated by the DPoE System. The DPoE System conveys this nonce to the DPoE ONU in an EAP-TLS Request PDU in the `client_key_exchange` TLV. The DPoE ONU decrypts the response with its private key, returning the cleartext value to the DPoE System in an EAP-TLS Request PDU in the `client_key_exchange` TLV. The DPoE System MUST validate that the response matches the original token in

addition to validating the X.509 certificate, the deny service (a PoE ONU that does not return the correct nonce even if the certificate is valid. See Figure 15 and Figure 16.

Superseded

Superseded

14 SECURE SOFTWARE DOWNLOAD (SSD)

The DPoE Network differs from DOCSIS in that many of the functions of the CM in DOCSIS are emulated in the DPoE System through the vCM. The implementation of SSD is thus split into two parts: one from the OSS to the DPoE System across the D interface, and the second from the DPoE System to DPoE ONU across the TU interface.

14.1 Secure File Transfer across the D Interface

SSD is used to transfer a new executable image to the DPoE ONU. The DPoE System **MUST** support SSD and Code Verification Certificates (CVCs) from the OSS to the DPoE System across the D interface.

This executable image is subject to code verification with a CVC. Vendors **MUST** digitally sign their code images with the Mfg CVC, as described in [SECv3.0]. Service providers **MAY** digitally co-sign vendor code images with the operator's CVC. The public key of the DPoE Root CA certificate is installed on the DPoE System for use in validating CVCs.

The DPoE System validates the CVCs for a DPoE ONU upgrade according to the requirements in [SECv3.0] for certificate content and time varying controls, except for using the DPoE certificate chain instead of the DOCSIS chain

14.2 Secure File Transfer across the TU Interface

The DPoE System **MUST** support transmission of any executable image (code) files over the TU interface. Since downstream encryption is required to be enabled across the TU interface, all file transfers, including the executable image (code) file transfer, are encrypted on the data path to a particular DPoE ONU. The DPoE System **MUST** transmit executable image files only after encryption is enabled. The DPoE System **MUST** initiate file transfer only from the DPoE System in accordance with the transfer protocol defined in [DPoE-SP-OAMv1.0].

A DPoE ONU does not initiate a file transfer by requesting a file. A file transfer can only be initiated from the DPoE System. Because DPoE OAM is a controlled channel and such frames cannot be injected from any source other than the DPoE System, this requirement means that the DPoE System and only the DPoE System initiates a file transfer.

Appendix I

Superseded

Example Frames

I.1 AES 128 CFB Encrypted Frame

Initial Vector (Hex):

30 31 32 33 34 35 36 37
38 39 3a 3b 8e 3e 5a ff

Key (Hex):

2b 7e 15 16 28 ae d2 a6
ab f7 15 88 09 cf 4f 3c

Plain Text (Hex):

DA -> 01 00 ff ff ff ff 42 43
44 45 46 47 48 49 4a 4b
4c 4d 4d 4f 50 51 52 53
54 55 56 57 58 59 5a 5b
5c 5d 5e 5f 60 61 62 63
64 65 66 67 68 69 6a 6b
6c 6d 6e 6f 70 71 72 73
74 75 76 77 91 73 1b 29 <- FCS

Cipher Text (Hex):

DA -> a4 7c a2 de 9f 4d ba f4
db ff 7d bd be 8b ed 72
78 fe 3c 5e 22 a8 84 8f
e3 e2 d4 8b 46 96 2b ab
4e cb 93 9c 62 b9 90 a7
8f 0c a6 6a 2c 31 38 be
8b 6e 9d 84 d9 c2 ff 04
e0 c3 34 46 96 c8 33 ba <- FCS

Superseded

Appendix II Reference AES Implementation (C programming language)

```

/**
 * rijndael-alg-fst.c
 *
 * @version 3.0 (December 2000)
 *
 * Optimised ANSI C code for the Rijndael cipher (now AES)
 *
 * @author Vincent Rijmen <vincent.rijmen@esat.kuleuven.ac.be>
 * @author Antoon Bosselaers <antoon.bosselaers@esat.kuleuven.ac.be>
 * @author Paulo Barreto <paulo.barreto@terra.com.br>
 *
 * This code is hereby placed in the public domain.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHORS 'AS IS' AND ANY EXPRESS
 * OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
 * BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
 * OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

#include <assert.h>
#include <stdlib.h>
#include "Python.h"

#define MODULE_NAME AES
#define BLOCK_SIZE 16
#define KEY_SIZE 0

#define MAXKC (256/32)
#define MAXKB (256/8)
#define MAXNR 14

typedef unsigned char    u8;
typedef unsigned short   u16;
typedef unsigned int     u32;

typedef struct {
    u32 ek[ 4*(MAXNR+1) ];
    u32 dk[ 4*(MAXNR+1) ];
    int rounds;
} block_state;

void rijndaelEncrypt(u32 rk[/4*(Nr + 1)*/], int Nr, const u8 pt[16], u8 ct[16]);
void rijndaelDecrypt(u32 rk[/4*(Nr + 1)*/], int Nr, const u8 ct[16], u8 pt[16]);

#ifdef INTERMEDIATE_VALUE_KAT
void rijndaelEncryptRound(const u32 rk[/4*(Nr + 1)*/], int Nr, u8 block[16], int rounds);
void rijndaelDecryptRound(const u32 rk[/4*(Nr + 1)*/], int Nr, u8 block[16], int rounds);
#endif /* INTERMEDIATE_VALUE_KAT */

/*
Te0[x] = S [x].[02, 01, 01, 03];
Te1[x] = S [x].[03, 02, 01, 01];
Te2[x] = S [x].[01, 03, 02, 01];
Te3[x] = S [x].[01, 01, 03, 02];

```

Superseded

```
Te4[x] = S [x].[01, 01, 01, 01];
Td0[x] = Si[x].[0e, 09, 0d, 0b];
Td1[x] = Si[x].[0b, 0e, 09, 0d];
Td2[x] = Si[x].[0d, 0b, 0e, 09];
Td3[x] = Si[x].[09, 0d, 0b, 0e];
Td4[x] = Si[x].[01, 01, 01, 01];
*/

static const u32 Te0[256] = {
    0xc66363a5U, 0xf87c7c84U, 0xee777799U, 0xf67b7b8dU,
    0xffff2f20dU, 0xd66b6bbdU, 0xde6f6fb1U, 0x91c5c554U,
    0x60303050U, 0x02010103U, 0xce6767a9U, 0x562b2b7dU,
    0xe7fefe19U, 0xb5d7d762U, 0x4dababe6U, 0xec76769aU,
    0x8fcaca45U, 0x1f82829dU, 0x89c9c940U, 0xfa7d7d87U,
    0xeffafa15U, 0xb25959ebU, 0x8e4747c9U, 0xfbfbf0f0bU,
    0x41adadecU, 0xb3d4d467U, 0x5fa2a2fdU, 0x45afafeaU,
    0x239c9cbfU, 0x53a4a4f7U, 0xe4727296U, 0x9bc0c05bU,
    0x75b7b7c2U, 0xe1fdfd1cU, 0x3d9393aeU, 0x4c26266aU,
    0x6c36365aU, 0x7e3f3f41U, 0xf5f7f702U, 0x83cccc4fU,
    0x6834345cU, 0x51a5a5f4U, 0xd1e5e534U, 0xf9f1f108U,
    0xe2717193U, 0xabd8d873U, 0x62313153U, 0x2a15153fU,
    0x0804040cU, 0x95c7c752U, 0x46232365U, 0x9dc3c35eU,
    0x30181828U, 0x379696a1U, 0x0a05050fU, 0x2f9a9ab5U,
    0x0e070709U, 0x24121236U, 0x1b80809bU, 0xdf2e2e23dU,
    0xcdeb26U, 0x4e272769U, 0x7fb2b2cdU, 0xea75759fU,
    0x1209091bU, 0x1d83839eU, 0x582c2c74U, 0x341a1a2eU,
    0x361b1b2dU, 0xdc6e6eb2U, 0xb45a5aeeU, 0x5ba0a0fbU,
    0xa45252f6U, 0x763b3b4dU, 0xb7d6d661U, 0x7db3b3ceU,
    0x5229297bU, 0xdde3e33eU, 0x5e2f2f71U, 0x13848497U,
    0xa65353f5U, 0xb9d1d168U, 0x00000000U, 0xc1eded2cU,
    0x40202060U, 0xe3fcfc1fU, 0x79b1b1c8U, 0xb65b5bedU,
    0xd46a6abeU, 0x8dc6c646U, 0x67bebed9U, 0x7239394bU,
    0x944a4adeU, 0x984c4cd4U, 0xb05858e8U, 0x85cfcf4aU,
    0xbbd0d06bU, 0xc5efef2aU, 0x4faaaae5U, 0xedfbfb16U,
    0x864343c5U, 0x9a4d4dd7U, 0x66333355U, 0x11858594U,
    0x8a4545cfU, 0xe9f9f910U, 0x04020206U, 0xfe7f7f81U,
    0xa05050f0U, 0x783c3c44U, 0x259f9fbaU, 0x4ba8a8e3U,
    0xa25151f3U, 0x5da3a3feU, 0x804040c0U, 0x058f8f8aU,
    0x3f9292adU, 0x219d9dbcU, 0x70383848U, 0xf1f5f504U,
    0x63bcbcdfU, 0x77b6b6c1U, 0xafdada75U, 0x42212163U,
    0x20101030U, 0xe5ffff1aU, 0xfdf3f30eU, 0xbfd2d26dU,
    0x81cdcd4cU, 0x180c0c14U, 0x26131335U, 0xc3ec2cfU,
    0xbe5f5fe1U, 0x359797a2U, 0x884444ccU, 0x2e171739U,
    0x93c4c457U, 0x55a7a7f2U, 0xfc7e7e82U, 0x7a3d3d47U,
    0xc86464acU, 0xba5d5de7U, 0x3219192bU, 0xe6737395U,
    0xc06060a0U, 0x19818198U, 0x9e4f4fd1U, 0xa3dc3dc7fU,
    0x44222266U, 0x542a2a7eU, 0x3b9090abU, 0x0b888883U,
    0x8c4646caU, 0xc7eeee29U, 0x6bb8b8d3U, 0x2814143cU,
    0xa7dede79U, 0xbc5e5ee2U, 0x160b0b1dU, 0xaddbdb76U,
    0xdbe0e03bU, 0x64323256U, 0x743a3a4eU, 0x140a0a1eU,
    0x924949dbU, 0x0c06060aU, 0x4824246cU, 0xb85c5ce4U,
    0x9fc2c25dU, 0xbdd3d36eU, 0x43acacefU, 0xc46262a6U,
    0x399191a8U, 0x319595a4U, 0xd3e4e437U, 0xf279798bU,
    0xd5e7e732U, 0x8bc8c843U, 0x6e373759U, 0xda6d6db7U,
    0x018d8d8cU, 0xb1d5d564U, 0x9c4e4ed2U, 0x49a9a9e0U,
    0xd86c6cb4U, 0xac5656faU, 0xf3f4f407U, 0xcfeaea25U,
    0xca6565afU, 0xf47a7a8eU, 0x47aeae9U, 0x10080818U,
    0x6fbabad5U, 0xf0787888U, 0x4a25256fU, 0x5c2e2e72U,
    0x381c1c24U, 0x57a6a6f1U, 0x73b4b4c7U, 0x97c6c651U,
    0xcbe8e823U, 0xa1dddd7cU, 0xe874749cU, 0x3e1f1f21U,
    0x964b4bddU, 0x61bdbddcU, 0xd8b8b86U, 0xf8a8a85U,
    0xe0707090U, 0x7c3e3e42U, 0x71b5b5c4U, 0xcc6666aaU,
    0x904848d8U, 0x06030305U, 0xf7f6f601U, 0xc0e0e012U,
    0xc26161a3U, 0x6a35355fU, 0xae5757f9U, 0x69b9b9d0U,
    0x17868691U, 0x99c1c158U, 0x3a1d1d27U, 0x279e9eb9U,
    0xd9e1e138U, 0xebf8f813U, 0x2b9898b3U, 0x22111133U,
```

Superseded

```
0xd26969bbU, 0xaad9c97U, 0x17813e89U, 0x312401aU,  
0x2d9b9bbU, 0x31e1e2U, 0x5878792U, 0xc2e9e2U,  
0x87cece49U, 0xaa5555fU, 0x50282878U, 0xa501df7aU,  
0x038c8c8fU, 0x59a1a1f8U, 0x09898980U, 0x1a0d0d17U,  
0x65bfbfdaU, 0xd7e6e631U, 0x844242c6U, 0xd06868b8U,  
0x824141c3U, 0x299999b0U, 0x5a2d2d77U, 0x1e0f0f11U,  
0x7bb0b0cbU, 0xa85454fcU, 0x6dbbbb6U, 0x2c16163aU,  
};  
static const u32 Te1[256] = {  
0xa5c66363U, 0x84f87c7cU, 0x99ee7777U, 0x8df67b7bU,  
0x0dfff2f2U, 0xbdd66b6bU, 0xb1de6f6fU, 0x5491c5c5U,  
0x50603030U, 0x03020101U, 0xa9ce6767U, 0x7d562b2bU,  
0x19e7efefU, 0x62b5d7d7U, 0xe64dababU, 0x9aec7676U,  
0x458fcacaU, 0x9d1f8282U, 0x4089c9c9U, 0x87fa7d7dU,  
0x15effafaU, 0xebb25959U, 0xc98e4747U, 0x0bfbf0f0U,  
0xec41adadU, 0x67b3d4d4U, 0xfd5fa2a2U, 0xea45afafU,  
0xbf239c9cU, 0xf753a4a4U, 0x96e47272U, 0x5b9bc0c0U,  
0xc275b7b7U, 0x1ce1fdfdU, 0xae3d9393U, 0x6a4c2626U,  
0x5a6c3636U, 0x417e3f3fU, 0x02f5f7f7U, 0x4f83ccccU,  
0x5c683434U, 0xf451a5a5U, 0x34d1e5e5U, 0x08f9f1f1U,  
0x93e27171U, 0x73abd8d8U, 0x53623131U, 0x3f2a1515U,  
0x0c080404U, 0x5295c7c7U, 0x65462323U, 0x5e9dc3c3U,  
0x28301818U, 0xa1379696U, 0x0f0a0505U, 0xb52f9a9aU,  
0x090e0707U, 0x36241212U, 0x9b1b8080U, 0x3ddfe2e2U,  
0x26cdebebU, 0x694e2727U, 0xcd7fb2b2U, 0x9fea7575U,  
0x1b120909U, 0x9e1d8383U, 0x74582c2cU, 0x2e341a1aU,  
0x2d361b1bU, 0xb2dc6e6eU, 0xeeb45a5aU, 0xfb5ba0a0U,  
0xf6a45252U, 0x4d763b3bU, 0x61b7d6d6U, 0xce7db3b3U,  
0x7b522929U, 0x3edde3e3U, 0x715e2f2fU, 0x97138484U,  
0xf5a65353U, 0x68b9d1d1U, 0x00000000U, 0x2cc1ededU,  
0x60402020U, 0x1fe3fcfcU, 0xc879b1b1U, 0xedb65b5bU,  
0xbcd46a6aU, 0x468dcbbcU, 0xd967bebeU, 0x4b723939U,  
0xde944a4aU, 0xd4984c4cU, 0xe8b05858U, 0xa85cfcfcU,  
0x6bbbd0d0U, 0x2ac5efefU, 0xe54faaaaU, 0x16edfbfbU,  
0xc5864343U, 0xd79a4d4dU, 0x55663333U, 0x94118585U,  
0xcf8a4545U, 0x10e9f9f9U, 0x06040202U, 0x81fe7f7fU,  
0xf0a05050U, 0x44783c3cU, 0xba259f9fU, 0xe34ba8a8U,  
0xf3a25151U, 0xfe5da3a3U, 0xc8040404U, 0xa8058f8fU,  
0xad3f9292U, 0xbc219d9dU, 0x48703838U, 0x04f1f5f5U,  
0xdf63bcbcbU, 0xc177b6b6U, 0x75afdadaU, 0x63422121U,  
0x30201010U, 0x1ae5ffffU, 0x0efdf3f3U, 0x6dbfd2d2U,  
0x4c81cdcdU, 0x14180c0cU, 0x35261313U, 0x2fc3ec3cU,  
0xe1be5f5fU, 0xa2359797U, 0xcc884444U, 0x392e1717U,  
0x5793c4c4U, 0xf255a7a7U, 0x82fc7e7eU, 0x477a3d3dU,  
0xacc86464U, 0xe7ba5d5dU, 0x2b321919U, 0x95e67373U,  
0xa0c06060U, 0x98198181U, 0xd19e4f4fU, 0x7fa3dcdcdU,  
0x66442222U, 0x7e542a2aU, 0xab3b9090U, 0x830b8888U,  
0xca8c4646U, 0x29c7eeeeU, 0xd36bb8b8U, 0x3c281414U,  
0x79a7dedeU, 0xe2bc5e5eU, 0x1d160b0bU, 0x76addbdbU,  
0x3dbde0e0U, 0x56643232U, 0x4e743a3aU, 0x1e140a0aU,  
0xdb924949U, 0x0a0c0606U, 0x6c482424U, 0xe4b85c5cU,  
0x5d9fc2c2U, 0x6ebdd3d3U, 0xef43acacU, 0xa6c46262U,  
0xa8399191U, 0xa4319595U, 0x37d3e4e4U, 0x8bf27979U,  
0x32d5e7e7U, 0x438bc8c8U, 0x596e3737U, 0xb7da6d6dU,  
0x8c018d8dU, 0x64b1d5d5U, 0xd29c4e4eU, 0xe049a9a9U,  
0xb4d86c6cU, 0xfaac5656U, 0x07f3f4f4U, 0x25cfeaeaU,  
0xafca6565U, 0x8ef47a7aU, 0xe947aeaeU, 0x18100808U,  
0xd56fbabaU, 0x88f07878U, 0x6f4a2525U, 0x725c2e2eU,  
0x24381c1cU, 0xf157a6a6U, 0xc773b4b4U, 0x5197c6c6U,  
0x23cbe8e8U, 0x7ca1ddddU, 0x9ce87474U, 0x213e1f1fU,  
0xdd964b4bU, 0xdc61bdbdU, 0x860d8b8bU, 0x850f8a8aU,  
0x90e07070U, 0x427c3e3eU, 0xc471b5b5U, 0xaacc6666U,  
0xd8904848U, 0x05060303U, 0x01f7f6f6U, 0x121c0e0eU,  
0xa3c26161U, 0x5f6a3535U, 0xf9ae5757U, 0xd069b9b9U,  
0x91178686U, 0x5899c1c1U, 0x273a1d1dU, 0xb9279e9eU,  
0x38d9e1e1U, 0x13ebf8f8U, 0xb32b9898U, 0x33221111U,  
0xbbd26969U, 0x70a9d9d9U, 0x89078e8eU, 0xa7339494U,
```

Superseded

```
0xb62d9b9bU, 0x213c1e1eU, 0x121f3737U, 0x21e0e1e1U,  
0x4987cecU, 0xf9aa955U, 0x85c2828U, 0x71a5d5dU,  
0x8f038c8cU, 0xf859a1aU, 0x80098989U, 0x171a0d0dU,  
0xda65bfbfU, 0x31d7e6e6U, 0xc6844242U, 0xb8d06868U,  
0xc3824141U, 0xb0299999U, 0x775a2d2dU, 0x111e0f0fU,  
0xcb7bb0b0U, 0xfca85454U, 0xd66dbbbbU, 0x3a2c1616U,  
};  
static const u32 Te2[256] = {  
0x63a5c663U, 0x7c84f87cU, 0x7799ee77U, 0x7b8df67bU,  
0xf20dfff2U, 0x6bbdd66bU, 0x6fb1de6fU, 0xc55491c5U,  
0x30506030U, 0x01030201U, 0x67a9ce67U, 0x2b7d562bU,  
0xfe19e7feU, 0xd762b5d7U, 0xab64dabU, 0x769aec76U,  
0xca458fcaU, 0x829d1f82U, 0xc94089c9U, 0x7d87fa7dU,  
0xfa15effaU, 0x59ebb259U, 0x47c98e47U, 0xf00bfbf0U,  
0xadec41adU, 0xd467b3d4U, 0xa2fd5fa2U, 0xafea45afU,  
0x9cbf239cU, 0xa4f753a4U, 0x7296e472U, 0xc05b9bc0U,  
0xb7c275b7U, 0xfd1ce1fdU, 0x93ae3d93U, 0x266a4c26U,  
0x365a6c36U, 0x3f417e3fU, 0xf702f5f7U, 0xcc4f83ccU,  
0x345c6834U, 0xa5f451a5U, 0xe534d1e5U, 0xf108f9f1U,  
0x7193e271U, 0xd873abd8U, 0x31536231U, 0x153f2a15U,  
0x040c0804U, 0xc75295c7U, 0x23654623U, 0xc35e9dc3U,  
0x18283018U, 0x96a13796U, 0x050f0a05U, 0x9ab52f9aU,  
0x07090e07U, 0x12362412U, 0x809b1b80U, 0xe23ddf2U,  
0xeb26cdebU, 0x27694e27U, 0xb2cd7fb2U, 0x759fea75U,  
0x091b1209U, 0x839e1d83U, 0x2c74582cU, 0x1a2e341aU,  
0x1b2d361bU, 0x6eb2dc6eU, 0x5aeeb45aU, 0xa0fb5ba0U,  
0x52f6a452U, 0x3b4d763bU, 0xd661b7d6U, 0xb3ce7db3U,  
0x297b5229U, 0xe33edde3U, 0x2f715e2fU, 0x84971384U,  
0x53f5a653U, 0xd168b9d1U, 0x00000000U, 0xed2cc1edU,  
0x20604020U, 0xfc1fe3fcU, 0xb1c879b1U, 0x5bedb65bU,  
0x6abed46aU, 0xcb468dcbU, 0xbed967beU, 0x394b7239U,  
0x4ade944aU, 0x4cd4984cU, 0x58e8b058U, 0xcf4a85cfU,  
0xd06bbbd0U, 0xef2ac5efU, 0xaae54faaU, 0xfb16edfbU,  
0x43c58643U, 0x4dd79a4dU, 0x33556633U, 0x85941185U,  
0x45cf8a45U, 0xf910e9f9U, 0x02060402U, 0x7f81fe7fU,  
0x50f0a050U, 0x3c44783cU, 0x9fba259fU, 0xa8e34ba8U,  
0x51f3a251U, 0xa3fe5da3U, 0x40c08040U, 0x8f8a058fU,  
0x92ad3f92U, 0x9dbc219dU, 0x38487038U, 0xf504f1f5U,  
0xbcdf63bcU, 0xb6c177b6U, 0xda75afdaU, 0x21634221U,  
0x10302010U, 0xff1ae5ffU, 0xf30efdf3U, 0xd26dbfd2U,  
0xcd4c81cdU, 0x0c14180cU, 0x13352613U, 0xec2fc3ecU,  
0x5fe1be5fU, 0x97a23597U, 0x44cc8844U, 0x17392e17U,  
0xc45793c4U, 0xa7f255a7U, 0x7e82fc7eU, 0x3d477a3dU,  
0x64acc864U, 0x5de7ba5dU, 0x192b3219U, 0x7395e673U,  
0x60a0c060U, 0x81981981U, 0x4fd19e4fU, 0xdc7fa3dcU,  
0x22664422U, 0x2a7e542aU, 0x90ab3b90U, 0x88830b88U,  
0x46ca8c46U, 0xee29c7eeU, 0xb8d36bb8U, 0x143c2814U,  
0xde79a7deU, 0x5ee2bc5eU, 0xb1d160b1U, 0xdb76addbU,  
0xe03bdbe0U, 0x32566432U, 0x3a4e743aU, 0x0a1e140aU,  
0x49db9249U, 0x060a0c06U, 0x246c4824U, 0x5ce4b85cU,  
0xc25d9fc2U, 0xd36ebdd3U, 0xacef43acU, 0x62a6c462U,  
0x91a83991U, 0x95a43195U, 0xe437d3e4U, 0x798bf279U,  
0xe732de7U, 0xc8438bc8U, 0x37596e37U, 0x6db7da6dU,  
0x8d8c018dU, 0xd564b1d5U, 0x4ed29c4eU, 0xa9e049a9U,  
0x6cb4d86cU, 0x56faac56U, 0xf407f3f4U, 0xea25cfeaU,  
0x65afca65U, 0x7a8ef47aU, 0xae947aeU, 0x08181008U,  
0xbad56fbaU, 0x7888f078U, 0x256f4a25U, 0x2e725c2eU,  
0x1c24381cU, 0xa6f157a6U, 0xb4c773b4U, 0xc65197c6U,  
0xe823cbe8U, 0xdd7ca1ddU, 0x749ce874U, 0x1f213e1fU,  
0x4bdd964bU, 0xbddc61bdU, 0x8b860d8bU, 0x8a850f8aU,  
0x7090e070U, 0x3e427c3eU, 0xb5c471b5U, 0x66aacc66U,  
0x48d89048U, 0x03050603U, 0xf601f7f6U, 0x0e121c0eU,  
0x61a3c261U, 0x355f6a35U, 0x57f9ae57U, 0xb9d069b9U,  
0x86911786U, 0xc15899c1U, 0x1d273a1dU, 0x9eb9279eU,  
0xe138d9e1U, 0xf813ebf8U, 0x98b32b98U, 0x11332211U,  
0x69bbd269U, 0xd970a9d9U, 0x8e89078eU, 0x94a73394U,  
0x9bb62d9bU, 0x1e223c1eU, 0x87921587U, 0xe920c9e9U,
```

Superseded

```

0xce4987ceU, 0x5ffa55U, 0x2875628U, 0xd7aa0dU,
0x8c8f038U, 0xaf899aU, 0x198c9939U, 0x017170U,
0xbfd65b5U, 0xe631d7eU, 0x42c68442U, 0x68b8d068U,
0x41c38241U, 0x99b02993U, 0x2d775a2dU, 0x0f111e0fU,
0xb0cb7bb0U, 0x54fca854U, 0xbbd66dbbU, 0x163a2c16U,
};
static const u32 Te3[256] = {

```

```

0x6363a5c6U, 0x7c7c84f8U, 0x777799eeU, 0x7b7b8df6U,
0xf2f20dffU, 0x6b6bddd6U, 0x6f6fb1deU, 0xc5c55491U,
0x30305060U, 0x01010302U, 0x6767a9ceU, 0x2b2b7d56U,
0xfefe19e7U, 0xd7d762b5U, 0xababe64dU, 0x76769aecU,
0xcaca458fU, 0x82829d1fU, 0xc9c94089U, 0x7d7d87faU,
0xfafa15efU, 0x5959ebb2U, 0x4747c98eU, 0xf0f00bfbU,
0xadadec41U, 0xd4d467b3U, 0xa2a2fd5fU, 0xafafea45U,
0x9c9cbf23U, 0xa4a4f753U, 0x727296e4U, 0xc0c05b9bU,
0xb7b7c275U, 0xfdfd1ce1U, 0x9393ae3dU, 0x26266a4cU,
0x36365a6cU, 0x3f3f417eU, 0xf7f702f5U, 0xcccc4f83U,
0x34345c68U, 0xa5a5f451U, 0xe5e534d1U, 0xf1f108f9U,
0x717193e2U, 0xd8d873abU, 0x31315362U, 0x15153f2aU,
0x04040c08U, 0xc7c75295U, 0x23236546U, 0xc3c35e9dU,
0x18182830U, 0x9696a137U, 0x05050f0aU, 0x9a9ab52fU,
0x0707090eU, 0x12123624U, 0x80809b1bU, 0xe2e23ddfU,
0xebeb26cdU, 0x2727694eU, 0xb2b2cd7fU, 0x75759feaU,
0x09091b12U, 0x83839e1dU, 0x2c2c7458U, 0x1a1a2e34U,
0x1b1b2d36U, 0x6e6eb2dcU, 0x5a5aeeb4U, 0xa0a0fb5bU,
0x5252f6a4U, 0x3b3b4d76U, 0xd6d661b7U, 0xb3b3ce7dU,
0x29297b52U, 0xe3e33eddU, 0x2f2f715eU, 0x84849713U,
0x5353f5a6U, 0xd1d168b9U, 0x00000000U, 0xeded2cc1U,
0x20206040U, 0xfcfc1fe3U, 0xb1b1c879U, 0x5b5bedb6U,
0x6a6abed4U, 0xcbcb468dU, 0xbeced967U, 0x39394b72U,
0x4a4ade94U, 0x4c4cd498U, 0x5858e8b0U, 0xcfcf4a85U,
0xd0d066bbU, 0xefef2ac5U, 0xaaaae54fU, 0xfbfb16edU,
0x4343c586U, 0x4d4dd79aU, 0x33335566U, 0x85859411U,
0x4545cf8aU, 0xf9f910e9U, 0x02020604U, 0x7f7f81feU,
0x5050f0a0U, 0x3c3c4478U, 0x9f9fba25U, 0xa8a8e34bU,
0x5151f3a2U, 0xa3a3fe5dU, 0x4040c080U, 0x8f8f8a05U,
0x9292ad3fU, 0x9d9dbcd21U, 0x38384870U, 0xf5f504f1U,
0xbcbcbdf63U, 0xb6b6c177U, 0xdada75afU, 0x21216342U,
0x10103020U, 0xffff1ae5U, 0xf3f30efdU, 0xd2d26dbfU,
0xcdcd4c81U, 0x0c0c1418U, 0x13133526U, 0xeccec2fc3U,
0x5f5fe1beU, 0x9797a235U, 0x4444cc88U, 0x1717392eU,
0xc4c45f93U, 0xa7a7f255U, 0x7e7e82fcU, 0x3d3d477aU,
0x6464acc8U, 0x5d5de7baU, 0x19192b32U, 0x737395e6U,
0x6060a0c0U, 0x81819819U, 0x4f4fd19eU, 0xdcdc7fa3U,
0x22226644U, 0x2a2a7e54U, 0x9090ab3bU, 0x8888830bU,
0x4646ca8cU, 0xeeee29c7U, 0xb8b8d36bU, 0x14143c28U,
0xdede79a7U, 0x5e5ee2bcU, 0x0b0b1d16U, 0xdbdb76adU,
0xe0e03bdbU, 0x32325664U, 0x3a3a4e74U, 0x0a0a1e14U,
0x4949db92U, 0x06060a0cU, 0x24246c48U, 0x5c5c4b8U,
0xc2c25d9fU, 0xd3d36ebdU, 0xacacef43U, 0x6262a6c4U,
0x9191a839U, 0x9595a431U, 0xe4e437d3U, 0x79798bf2U,
0xe7e732d5U, 0xc8c8438bU, 0x3737596eU, 0x6d6db7daU,
0x8d8d8c01U, 0xd5d564b1U, 0x4e4ed29cU, 0xa9a9e049U,
0x6c6cb4d8U, 0x5656faacU, 0xf4f407f3U, 0xeaea25cfU,
0x6565afcaU, 0x7a7a8ef4U, 0xaeae947U, 0x08081810U,
0xbabad56fU, 0x787888f0U, 0x25256f4aU, 0x2e2e725cU,
0x1c1c2438U, 0xa6a6f157U, 0xb4b4c773U, 0xc6c65197U,
0xe8e823cbU, 0xdddd7ca1U, 0x74749ce8U, 0x1f1f213eU,
0x4b4bdd96U, 0xbdbddc61U, 0x8b8b860dU, 0x8a8a850fU,
0x707090e0U, 0x3e3e427cU, 0xb5b5c471U, 0x6666aaccU,
0x4848d890U, 0x03030506U, 0xf6f601f7U, 0x0e0e121cU,
0x6161a3c2U, 0x35355f6aU, 0x5757f9aeU, 0xb9b9d069U,
0x86869117U, 0xc1c15899U, 0x1d1d273aU, 0x9e9eb927U,
0xe1e138d9U, 0xf8f813ebU, 0x9898b32bU, 0x11113322U,
0x6969bbd2U, 0xd9d970a9U, 0x8e8e8907U, 0x9494a733U,
0x9b9bb62dU, 0x1e1e223cU, 0x87879215U, 0xe9e920c9U,

```

Superseded

```
0xcece4987U, 0x5551faU, 0x28287860U, 0xd4e7aU,
0x8c8c8f0U, 0xa1a135U, 0x98c8c9U, 0x0d171U,
0xbfbfda65U, 0xe6e631dU, 0x4242c684U, 0x6868b8d0U,
0x4141c382U, 0x9999b023U, 0x2d2d775aU, 0x0f0f111eU,
0xb0b0cb7bU, 0x5454fca8U, 0xbbbbd66dU, 0x16163a2cU,
};
static const u32 Te4[256] = {
0x63636363U, 0x7c7c7c7cU, 0x77777777U, 0x7b7b7b7bU,
0xf2f2f2f2U, 0x6b6b6b6bU, 0x6f6f6f6fU, 0xc5c5c5c5U,
0x30303030U, 0x01010101U, 0x67676767U, 0x2b2b2b2bU,
0xfefefefefU, 0xd7d7d7d7U, 0xababababU, 0x76767676U,
0xcacacacaU, 0x82828282U, 0xc9c9c9c9U, 0x7d7d7d7dU,
0xfafafafaU, 0x59595959U, 0x47474747U, 0xf0f0f0f0U,
0xadadadadU, 0xd4d4d4d4U, 0xa2a2a2a2U, 0xafafafafU,
0x9c9c9c9cU, 0xa4a4a4a4U, 0x72727272U, 0xc0c0c0c0U,
0xb7b7b7b7U, 0xfdfdfdfdfU, 0x93939393U, 0x26262626U,
0x36363636U, 0x3f3f3f3fU, 0xf7f7f7f7U, 0xc0c0c0c0U,
0x34343434U, 0xa5a5a5a5U, 0xe5e5e5e5U, 0xf1f1f1f1U,
0x71717171U, 0xd8d8d8d8U, 0x31313131U, 0x15151515U,
0x04040404U, 0xc7c7c7c7U, 0x23232323U, 0xc3c3c3c3U,
0x18181818U, 0x96969696U, 0x05050505U, 0x9a9a9a9aU,
0x07070707U, 0x12121212U, 0x80808080U, 0xe2e2e2e2U,
0xeb9eb9ebU, 0x27272727U, 0xb2b2b2b2U, 0x75757575U,
0x09090909U, 0x83838383U, 0x2c2c2c2cU, 0x1a1a1a1aU,
0x1b1b1b1bU, 0x6e6e6e6eU, 0x5a5a5a5aU, 0xa0a0a0a0U,
0x52525252U, 0x3b3b3b3bU, 0xd6d6d6d6U, 0xb3b3b3b3U,
0x29292929U, 0xe3e3e3e3U, 0x2f2f2f2fU, 0x84848484U,
0x53535353U, 0xd1d1d1d1U, 0x00000000U, 0xedededU,
0x20202020U, 0xfcfcfcfcU, 0xb1b1b1b1U, 0x5b5b5b5bU,
0x6a6a6a6aU, 0xcbcbcbcbU, 0xb9b9b9b9U, 0x39393939U,
0x4a4a4a4aU, 0x4c4c4c4cU, 0x58585858U, 0xcfcfcfcfU,
0xd0d0d0d0U, 0xfefefefefU, 0xaaaaaaaaU, 0xfbfbfbfbfU,
0x43434343U, 0x4d4d4d4dU, 0x33333333U, 0x85858585U,
0x45454545U, 0xf9f9f9f9U, 0x02020202U, 0x7f7f7f7fU,
0x50505050U, 0x3c3c3c3cU, 0x9f9f9f9fU, 0xa8a8a8a8U,
0x51515151U, 0xa3a3a3a3U, 0x40404040U, 0x8f8f8f8fU,
0x92929292U, 0x9d9d9d9dU, 0x38383838U, 0xf5f5f5f5U,
0xbcbcbcbcbU, 0xb6b6b6b6U, 0xdadadadaU, 0x21212121U,
0x10101010U, 0xffffffffU, 0xf3f3f3f3U, 0xd2d2d2d2U,
0xcdcdcdcdU, 0xc0c0c0c0U, 0x13131313U, 0xec9ec9ecU,
0x5f5f5f5fU, 0x97979797U, 0x44444444U, 0x17171717U,
0xc4c4c4c4U, 0xa7a7a7a7U, 0x7e7e7e7eU, 0x3d3d3d3dU,
0x64646464U, 0x5d5d5d5dU, 0x19191919U, 0x73737373U,
0x60606060U, 0x81818181U, 0x4f4f4f4fU, 0xdc9cdc9cU,
0x22222222U, 0x2a2a2a2aU, 0x90909090U, 0x88888888U,
0x46464646U, 0xe0e0e0e0U, 0xb8b8b8b8U, 0x14141414U,
0xd9d9d9d9U, 0x5e5e5e5eU, 0xb0b0b0b0U, 0xdbdbdbdbU,
0xe0e0e0e0U, 0x32323232U, 0x3a3a3a3aU, 0x0a0a0a0aU,
0x49494949U, 0x06060606U, 0x24242424U, 0x5c5c5c5cU,
0xc2c2c2c2U, 0xd3d3d3d3U, 0xacacacacU, 0x62626262U,
0x91919191U, 0x95959595U, 0xe4e4e4e4U, 0x79797979U,
0xe7e7e7e7U, 0xc8c8c8c8U, 0x37373737U, 0x6d6d6d6dU,
0x8d8d8d8dU, 0xd5d5d5d5U, 0x4e4e4e4eU, 0xa9a9a9a9U,
0x6c6c6c6cU, 0x56565656U, 0xf4f4f4f4U, 0xeaeaeaeaeU,
0x65656565U, 0x7a7a7a7aU, 0xaeaeaeaeU, 0x08080808U,
0xbababababU, 0x78787878U, 0x25252525U, 0x2e2e2e2eU,
0x1c1c1c1cU, 0xa6a6a6a6U, 0xb4b4b4b4U, 0xc6c6c6c6U,
0xe8e8e8e8U, 0xddddddddU, 0x74747474U, 0xf1f1f1f1U,
0x4b4b4b4bU, 0xbdbdbdbdbU, 0x8b8b8b8bU, 0x8a8a8a8aU,
0x70707070U, 0x3e3e3e3eU, 0xb5b5b5b5U, 0x66666666U,
0x48484848U, 0x03030303U, 0xf6f6f6f6U, 0x0e0e0e0eU,
0x61616161U, 0x35353535U, 0x57575757U, 0xb9b9b9b9U,
0x86868686U, 0xc1c1c1c1U, 0xd1d1d1d1U, 0x9e9e9e9eU,
0xe1e1e1e1U, 0xf8f8f8f8U, 0x98989898U, 0x11111111U,
0x69696969U, 0xd9d9d9d9U, 0x8e8e8e8eU, 0x94949494U,
0x9b9b9b9bU, 0x1e1e1e1eU, 0x87878787U, 0xe9e9e9e9U,
0xcecececeU, 0x55555555U, 0x28282828U, 0xdfdfdfdfU,
```

Superseded

```
0x8c8c8c8cU, 0xa1a1a1a1U, 0x20202020U, 0x00000000U,  
0xbfbfbfbfU, 0xe6e6e6e6U, 0x12424242U, 0x56565656U,  
0x41414141U, 0x99999999U, 0x2d2d2d2dU, 0xf0f0f0f0U,  
0xb0b0b0b0U, 0x54545454U, 0xb0b0b0b0U, 0x16161616U,  
};  
static const u32 Td0[256] = {  
0x51f4a750U, 0x7e416553U, 0x1a17a4c3U, 0x3a275e96U,  
0x3bab6bcbU, 0x1f9d45f1U, 0xacfa58abU, 0x4be30393U,  
0x2030fa55U, 0xad766df6U, 0x88cc7691U, 0xf5024c25U,  
0x4fe5d7fcU, 0xc52acbd7U, 0x26354480U, 0xb562a38fU,  
0xdeb15a49U, 0x25ba1b67U, 0x45ea0e98U, 0x5dfec0e1U,  
0xc32f7502U, 0x814cf012U, 0x8d4697a3U, 0x6bd3f9c6U,  
0x038f5fe7U, 0x15929c95U, 0xbf6d7aebU, 0x955259daU,  
0xd4be832dU, 0x587421d3U, 0x49e06929U, 0x8ec9c844U,  
0x75c2896aU, 0xf48e7978U, 0x99583e6bU, 0x27b971ddU,  
0xbee14fb6U, 0xf088ad17U, 0xc920ac66U, 0x7dce3ab4U,  
0x63df4a18U, 0xe51a3182U, 0x97513360U, 0x62537f45U,  
0xb16477e0U, 0xbb6bae84U, 0xfe81a01cU, 0xf9082b94U,  
0x70486858U, 0x8f45fd19U, 0x94de6c87U, 0x527bf8b7U,  
0xab73d323U, 0x724b02e2U, 0xe31f8f57U, 0x6655ab2aU,  
0xb2eb2807U, 0x2fb5c203U, 0x86c57b9aU, 0xd33708a5U,  
0x302887f2U, 0x23bfa5b2U, 0x02036abaU, 0xed16825cU,  
0x8acf1c2bU, 0xa779b492U, 0xf307f2f0U, 0x4e69e2a1U,  
0x65daf4cdU, 0x0605bed5U, 0xd134621fU, 0xc4a6fe8aU,  
0x342e539dU, 0xa2f355a0U, 0x058ae132U, 0xa4f6eb75U,  
0x0b83ec39U, 0x4060efaaU, 0x5e719f06U, 0xbd6e1051U,  
0x3e218af9U, 0x96dd063dU, 0xdd3e05aeU, 0x4de6bd46U,  
0x91548db5U, 0x71c45d05U, 0x0406d46fU, 0x605015ffU,  
0x1998fb24U, 0xd6bde997U, 0x894043ccU, 0x67d99e77U,  
0xb0e842bdU, 0x07898b88U, 0xe7195b38U, 0x79c8eedbU,  
0xa17c0a47U, 0x7c420fe9U, 0xf8841ec9U, 0x00000000U,  
0x09808683U, 0x322bed48U, 0x1e1170acU, 0x6c5a724eU,  
0xfd0efffbU, 0x0f853856U, 0x3daed51eU, 0x362d3927U,  
0x0a0fd964U, 0x685ca621U, 0x9b5b54d1U, 0x24362e3aU,  
0x0c0a67b1U, 0x9357e70fU, 0xb4ee96d2U, 0x1b9b919eU,  
0x80c0c54fU, 0x61dc20a2U, 0x5a774b69U, 0x1c121a16U,  
0xe293ba0aU, 0xc0a02ae5U, 0x3c22e043U, 0x121b171dU,  
0x0e090d0bU, 0xf28bc7adU, 0x2db6a8b9U, 0x141ea9c8U,  
0x57f11985U, 0xaf75074cU, 0xee99ddb0U, 0xa37f60fdU,  
0xf701269fU, 0x5c72f5bcU, 0x44663bc5U, 0x5bfb7e34U,  
0x8b432976U, 0xcb23c6dcU, 0xb6edfc68U, 0xb8e4f163U,  
0xd731dccau, 0x42638510U, 0x13972240U, 0x84c61120U,  
0x854a247dU, 0xd2bb3df8U, 0xae9f3211U, 0xc729a16dU,  
0x1d9e2f4bU, 0xdc230f3U, 0x0d8652ecU, 0x77c1e3d0U,  
0x2bb3166cU, 0xa970b999U, 0x119448faU, 0x47e96422U,  
0xa8fc8cc4U, 0xa0f03f1aU, 0x567d2cd8U, 0x223390efU,  
0x87494ec7U, 0xd938d1c1U, 0x8ccaa2feU, 0x98d40b36U,  
0xa6f581cfU, 0xa57ade28U, 0xdab78e26U, 0x3fadbfa4U,  
0x2c3a9de4U, 0x5078920dU, 0x6a5fcc9bU, 0x547e4662U,  
0xf68d13c2U, 0x90d8b8e8U, 0x2e39f75eU, 0x82c3aff5U,  
0x9f5d80beU, 0x69d0937cU, 0x6fd52da9U, 0xcf2512b3U,  
0xc8ac993bU, 0x10187da7U, 0xe89c636eU, 0xdb3bbb7bU,  
0xcd267809U, 0x6e5918f4U, 0xec9ab701U, 0x834f9aa8U,  
0xe6956e65U, 0xaaaffe67eU, 0x21bccf08U, 0xef15e8e6U,  
0xbae79bd9U, 0x4a6f36ceU, 0xea9f09d4U, 0x29b07cd6U,  
0x31a4b2afU, 0x2a3f2331U, 0xc6a59430U, 0x35a266c0U,  
0x744ebc37U, 0xfc82caa6U, 0xe090d0b0U, 0x33a7d815U,  
0xf104984aU, 0x41ecdaf7U, 0x7fcd500eU, 0x1791f62fU,  
0x764dd68dU, 0x43efb04dU, 0xccaa4d54U, 0xe49604dfU,  
0x9ed1b5e3U, 0x4c6a881bU, 0xc12c1fb8U, 0x4665517fU,  
0x9d5eea04U, 0x018c355dU, 0xfa877473U, 0xfb0b412eU,  
0xb3671d5aU, 0x92dbd252U, 0xe9105633U, 0x6dd64713U,  
0x9ad7618cU, 0x37a10c7aU, 0x59f8148eU, 0xeb133c89U,  
0xcea927eeU, 0xb761c935U, 0xe11ce5edU, 0x7a47b13cU,  
0x9cd2df59U, 0x55f2733fU, 0x1814ce79U, 0x73c737bfU,  
0x53f7cdeaU, 0x5ffdaa5bU, 0xdf3d6f14U, 0x7844db86U,  
0xcaaff381U, 0xb968c43eU, 0x3824342cU, 0xc2a3405fU,
```

Superseded

```
0x161dc372U, 0xcb0e2250U, 0x282f493bU, 0xf0d014U,  
0x39a8017U, 0x00c13dU, 0x18b7e49cU, 0x6556c49U,  
0x7bcb8461U, 0xd532b67U, 0x486c5c74U, 0xd0b85742U,  
};  
static const u32 Td1[256] = {  
0x5051f4a7U, 0x537e4165U, 0xc31a17a4U, 0x963a275eU,  
0xcb3bab6bU, 0xf11f9d45U, 0xabacfa58U, 0x934be303U,  
0x552030faU, 0xf6ad766dU, 0x9188cc76U, 0x25f5024cU,  
0xfc4fe5d7U, 0xd7c52acbU, 0x80263544U, 0x8fb562a3U,  
0x49deb15aU, 0x6725ba1bU, 0x9845ea0eU, 0xe15dfec0U,  
0x02c32f75U, 0x12814cf0U, 0xa38d4697U, 0xc66bd3f9U,  
0xe7038f5fU, 0x9515929cU, 0xebbf6d7aU, 0xda955259U,  
0x2dd4be83U, 0xd3587421U, 0x2949e069U, 0x448ec9c8U,  
0x6a75c289U, 0x78f48e79U, 0x6b99583eU, 0xdd27b971U,  
0xb6bee14fU, 0x17f088adU, 0x66c920acU, 0xb47dce3aU,  
0x1863df4aU, 0x82e51a31U, 0x60975133U, 0x4562537fU,  
0xe0b16477U, 0x84bb6baeU, 0x1cfe81a0U, 0x94f9082bU,  
0x58704868U, 0x198f45fdU, 0x8794de6cU, 0xb7527bf8U,  
0x23ab73d3U, 0xe2724b02U, 0x57e31f8fU, 0x2a6655abU,  
0x07b2eb28U, 0x032fb5c2U, 0x9a86c57bU, 0xa5d33708U,  
0xf2302887U, 0xb223bfa5U, 0xba02036aU, 0x5ced1682U,  
0x2b8acf1cU, 0x92a779b4U, 0xf0f307f2U, 0xa14e69e2U,  
0xcd65daf4U, 0xd50605beU, 0x1fd13462U, 0x8ac4a6feU,  
0x9d342e53U, 0xaa0a2f355U, 0x32058ae1U, 0x75a4f6ebU,  
0x390b83ecU, 0xaa4060efU, 0x065e719fU, 0x51bd6e10U,  
0xf93e218aU, 0x3d96dd06U, 0xaedd3e05U, 0x464de6bdU,  
0xb591548dU, 0x0571c45dU, 0x6f0406d4U, 0xff605015U,  
0x241998fbU, 0x97d6bde9U, 0xcc894043U, 0x7767d99eU,  
0xbdb0e842U, 0x8807898bU, 0x38e7195bU, 0xdb79c8eeU,  
0x47a17c0aU, 0xe97c420fU, 0xc9f8841eU, 0x00000000U,  
0x83098086U, 0x48322bedU, 0xac1e1170U, 0x4e6c5a72U,  
0xfbfdf0effU, 0x560f8538U, 0x1e3daed5U, 0x27362d39U,  
0x640a0dfdU, 0x21685ca6U, 0xd19b5b54U, 0x3a24362eU,  
0xb10c0a67U, 0x0f9357e7U, 0xd2b4ee96U, 0x9e1b9b91U,  
0x4f80c0c5U, 0xa261dc20U, 0x695a774bU, 0x161c121aU,  
0x0ae293baU, 0xe5c0a02aU, 0x433c22e0U, 0x1d121b17U,  
0x0b0e090dU, 0xadf28bc7U, 0xb92db6a8U, 0xc8141ea9U,  
0x8557f119U, 0x4caf7507U, 0xbbee99ddU, 0xfda37f60U,  
0x9ff70126U, 0xbc5c72f5U, 0xc544663bU, 0x345bfb7eU,  
0x768b4329U, 0xdcdb23c6U, 0x68b6edfcU, 0x63b8e4f1U,  
0xcad731dcU, 0x10426385U, 0x40139722U, 0x2084c611U,  
0x7d854a24U, 0xf8d2bb3dU, 0x11aef932U, 0x6dc729a1U,  
0x4b1d9e2fU, 0xf3dcb230U, 0xec0d8652U, 0xd077c1e3U,  
0x6c2bb316U, 0x99a970b9U, 0xfa119448U, 0x2247e964U,  
0xc4a8fc8cU, 0x1aa0f03fU, 0xd8567d2cU, 0xef223390U,  
0xc787494eU, 0xc1d938d1U, 0xfe8ccaa2U, 0x3698d40bU,  
0xcfa6f581U, 0x28a57adeU, 0x26dab78eU, 0xa43fadbffU,  
0xe42c3a9dU, 0x0d507892U, 0x9b6a5fccU, 0x62547e46U,  
0xc2f68d13U, 0xe890d8b8U, 0x5e2e39f7U, 0xf582c3afU,  
0xbe9f5d80U, 0x7c69d093U, 0xa96fd52dU, 0xb3cf2512U,  
0x3bc8ac99U, 0xa710187dU, 0x6ee89c63U, 0x7bdb3bbbU,  
0x09cd2678U, 0xf46e5918U, 0x01ec9ab7U, 0xa8834f9aU,  
0x65e6956eU, 0x7eaaaffe6U, 0x0821bccfU, 0xe6ef15e8U,  
0xd9bae79bU, 0xce4a6f36U, 0xd4ea9f09U, 0xd629b07cU,  
0xaf31a4b2U, 0x312a3f23U, 0x30c6a594U, 0xc035a266U,  
0x37744ebcU, 0xa6fc82caU, 0xb0e090d0U, 0x1533a7d8U,  
0x4af10498U, 0xf741ecdau, 0x0e7fcd50U, 0x2f1791f6U,  
0x8d764dd6U, 0x4d43efb0U, 0x54ccaa4dU, 0xdf49604U,  
0xe39ed1b5U, 0x1b4c6a88U, 0xb8c12c1fU, 0x7f466551U,  
0x049d5eeaU, 0x5d018c35U, 0x73fa8774U, 0x2efb0b41U,  
0x5ab3671dU, 0x5292dbd2U, 0x33e91056U, 0x136dd647U,  
0x8c9ad761U, 0x7a37a10cU, 0x8e59f814U, 0x89eb133cU,  
0xeecea927U, 0x35b761c9U, 0xede11ce5U, 0x3c7a47b1U,  
0x599cd2dfU, 0x3f55f273U, 0x791814ceU, 0xbf73c737U,  
0xea53f7cdU, 0x5b5ffdaaU, 0x14df3d6fU, 0x867844dbU,  
0x81caaff3U, 0x3eb968c4U, 0x2c382434U, 0x5fc2a340U,  
0x72161dc3U, 0x0cbce225U, 0x8b283c49U, 0x41ff0d95U,
```

Superseded

```
0x7139a801U, 0xd086cb4U, 0x2ed8b2e4U, 0x9664f1cU,
0x617bcb8U, 0x7d522bU, 0x74496c5cU, 0x4d0b95U,
};
static const u32 Td2[256] = {
0xa75051f4U, 0x65537e41U, 0xa4c31a17U, 0x5e963a27U,
0x6bcb3babU, 0x45f11f9dU, 0x58abacfaU, 0x03934be3U,
0xfa552030U, 0x6df6ad76U, 0x769188ccU, 0x4c25f502U,
0xd7fc4fe5U, 0xcdb7c52aU, 0x44802635U, 0xa38fb562U,
0x5a49deb1U, 0x1b6725baU, 0x0e9845eaU, 0xc0e15dfeU,
0x7502c32fU, 0xf012814cU, 0x97a38d46U, 0xf9c66bd3U,
0x5fe7038fU, 0x9c951592U, 0x7aebbf6dU, 0x59da9552U,
0x832dd4beU, 0x21d35874U, 0x692949e0U, 0xc8448ec9U,
0x896a75c2U, 0x7978f48eU, 0x3e6b9958U, 0x71dd27b9U,
0x4fb6bee1U, 0xad17f088U, 0xac66c920U, 0x3ab47dceU,
0x4a1863dfU, 0x3182e51aU, 0x33609751U, 0x7f456253U,
0x77e0b164U, 0xae84bb6bU, 0xa01cfe81U, 0x2b94f908U,
0x68587048U, 0xfd198f45U, 0x6c879adeU, 0xf8b7527bU,
0xd323ab73U, 0x02e2724bU, 0x8f57e31fU, 0xab2a6655U,
0x2807b2ebU, 0xc2032fb5U, 0x7b9a86c5U, 0x08a5d337U,
0x87f23028U, 0xa5b223bfU, 0x6aba0203U, 0x825ced16U,
0x1c2b8acfU, 0xb492a779U, 0xf2f0f307U, 0xe2a14e69U,
0xf4cd65daU, 0xbed50605U, 0x621fd134U, 0xfe8ac4a6U,
0x539d342eU, 0x55a0a2f3U, 0xe132058aU, 0xeb75a4f6U,
0xec390b83U, 0xefaa4060U, 0x9f065e71U, 0x1051bd6eU,

0x8af93e21U, 0x063d96ddU, 0x05aedd3eU, 0xbd464de6U,
0x8db59154U, 0x5d0571c4U, 0xd46f0406U, 0x15ff6050U,
0xfb241998U, 0xe997d6bdU, 0x43cc8940U, 0x9e7767d9U,
0x42bdb0e8U, 0x8b880789U, 0x5b38e719U, 0xeedb79c8U,
0x0a47a17cU, 0x0fe97c42U, 0x1ec9f884U, 0x00000000U,
0x86830980U, 0xed48322bU, 0x70ac1e11U, 0x724e6c5aU,
0xffffbf0eU, 0x38560f85U, 0xd51e3daeU, 0x3927362dU,
0xd9640a0fU, 0xa621685cU, 0x54d19b5bU, 0x2e3a2436U,
0x67b10c0aU, 0xe70f9357U, 0x96d2b4eeU, 0x919e1b9bU,
0xc54f80c0U, 0x20a261dcU, 0x4b695a77U, 0x1a161c12U,
0xba0ae293U, 0x2ae5c0a0U, 0xe0433c22U, 0x171d121bU,
0x0d0b0e09U, 0xc7adf28bU, 0xa8b92db6U, 0xa9c8141eU,
0x198557f1U, 0x074caf75U, 0xddbbbee9U, 0x60fda37fU,
0x269ff701U, 0xf5bc5c72U, 0x3bc54466U, 0x7e345bfbU,
0x29768b43U, 0xc6dccb23U, 0xfc68b6edU, 0xf163b8e4U,
0xdccad731U, 0x85104263U, 0x22401397U, 0x112084c6U,
0x247d854aU, 0x3df8d2bbU, 0x3211aef9U, 0xa16dc729U,
0x2f4b1d9eU, 0x30f3dcb2U, 0x52ec0d86U, 0xe3d077c1U,
0x166c2bb3U, 0xb999a970U, 0x48fa1194U, 0x642247e9U,
0x8cc4a8fcU, 0x3f1aa0f0U, 0x2cd8567dU, 0x90ef2233U,
0x4ec78749U, 0xd1c1d938U, 0xa2fe8ccaU, 0x0b3698d4U,
0x81cfa6f5U, 0xde28a57aU, 0x8e26dab7U, 0xbfa43fadU,
0x9de42c3aU, 0x920d5078U, 0xcc9b6a5fU, 0x4662547eU,
0x13c2f68dU, 0xb8e890d8U, 0xf75e2e39U, 0xaff582c3U,
0x80be9f5dU, 0x937c69d0U, 0x2da96fd5U, 0x12b3cf25U,
0x993bc8acU, 0x7da71018U, 0x636ee89cU, 0xbb7bdb3bU,
0x7809cd26U, 0x18f46e59U, 0xb701ec9aU, 0x9aa8834fU,
0x6e65e695U, 0xe67eaaafU, 0xcf0821bcU, 0xe8e6ef15U,
0x9bd9bae7U, 0x36ce4a6fU, 0x09d4ea9fU, 0x7cd629b0U,
0xb2af31a4U, 0x23312a3fU, 0x9430c6a5U, 0x66c035a2U,
0xbc37744eU, 0xcaa6fc82U, 0xd0b0e090U, 0xd81533a7U,
0x984af104U, 0xdaf741ecU, 0x500e7fcdU, 0xf62f1791U,
0xd68d764dU, 0xb04d43efU, 0x4d54ccaaU, 0x04dfe496U,
0xb5e39ed1U, 0x881b4c6aU, 0x1fb8c12cU, 0x517f4665U,
0xea049d5eU, 0x355d018cU, 0x7473fa87U, 0x412efb0bU,
0x1d5ab367U, 0xd25292dbU, 0x5633e910U, 0x47136dd6U,
0x618c9ad7U, 0x0c7a37a1U, 0x148e59f8U, 0x3c89eb13U,
0x27eecea9U, 0xc935b761U, 0xe5ede11cU, 0xb13c7a47U,
0xdf599cd2U, 0x733f55f2U, 0xce791814U, 0x37bf73c7U,
0xcdea53f7U, 0xaa5b5ffdu, 0x6f14df3dU, 0xdb867844U,
0xf381caafU, 0xc43eb968U, 0x342c3824U, 0x405fc2a3U,
0xc372161dU, 0x250cbce2U, 0x498b283cU, 0x9541ff0dU,
```

Superseded

```
0x017139a8U, 0xbde630aU, 0x40c1d804U, 0xc00c15fU,  
0x84617bcU, 0xb70053aU, 0x5c71486cU, 0x542d9bU,  
};  
static const u32 Td3[256] = {  
0xf4a75051U, 0x4165537eU, 0x17a4c31aU, 0x275e963aU,  
0xab6bcb3bU, 0x9d45f11fU, 0xfa58abacU, 0xe303934bU,  
0x30fa5520U, 0x766df6adU, 0xcc769188U, 0x024c25f5U,  
0xe5d7fc4fU, 0x2acbd7c5U, 0x35448026U, 0x62a38fb5U,  
0xb15a49deU, 0xba1b6725U, 0xea0e9845U, 0xfec0e15dU,  
0x2f7502c3U, 0x4cf01281U, 0x4697a38dU, 0xd3f9c66bU,  
0x8f5fe703U, 0x929c9515U, 0x6d7aebbfU, 0x5259da95U,  
0xbe832dd4U, 0x7421d358U, 0xe0692949U, 0xc9c8448eU,  
0xc2896a75U, 0x8e7978f4U, 0x583e6b99U, 0xb971dd27U,  
0xe14fb6beU, 0x88ad17f0U, 0x20ac66c9U, 0xce3ab47dU,  
0xdf4a1863U, 0x1a3182e5U, 0x51336097U, 0x537f4562U,  
0x6477e0b1U, 0x6bae84bbU, 0x81a01cfeU, 0x082b94f9U,  
0x48685870U, 0x45fd198fU, 0xde6c8794U, 0x7bf8b752U,  
0x73d323abU, 0x4b02e272U, 0x1f8f57e3U, 0x55ab2a66U,  
0xeb2807b2U, 0xb5c2032fU, 0xc57b9a86U, 0x3708a5d3U,  
0x2887f230U, 0xbfa5b223U, 0x036aba02U, 0x16825cedU,  
0xcfc1c2b8aU, 0x79b492a7U, 0x07f2f0f3U, 0x69e2a14eU,  
0xdaf4cd65U, 0x05bed506U, 0x34621fd1U, 0xa6fe8ac4U,  
0x2e539d34U, 0xf355a0a2U, 0x8ae13205U, 0xf6eb75a4U,  
0x83ec390bU, 0x60efaa40U, 0x719f065eU, 0x6e1051bdU,  
0x218af93eU, 0xdd063d96U, 0x3e05aeddU, 0xe6bd464dU,  
0x548db591U, 0xc45d0571U, 0x06d46f04U, 0x5015ff60U,  
0x98fb2419U, 0xbde997d6U, 0x4043cc89U, 0xd99e7767U,  
0xe842bdb0U, 0x898b8807U, 0x195b38e7U, 0xc8eedb79U,  
0x7c0a47a1U, 0x420fe97cU, 0x841ec9f8U, 0x00000000U,  
0x80868309U, 0x2bed4832U, 0x1170ac1eU, 0x5a724e6cU,  
0x0efffbfdU, 0x8538560fU, 0xaed51e3dU, 0x2d392736U,  
0x0fd9640aU, 0x5ca62168U, 0x5b54d19bU, 0x362e3a24U,  
0x0a67b10cU, 0x57e70f93U, 0xee96d2b4U, 0x9b919e1bU,  
0xc0c54f80U, 0xdc20a261U, 0x774b695aU, 0x121a161cU,  
0x93ba0ae2U, 0xa02ae5c0U, 0x22e0433cU, 0x1b171d12U,  
0x090d0b0eU, 0x8bc7adf2U, 0xb6a8b92dU, 0x1ea9c814U,  
0xf1198557U, 0x75074cafU, 0x99ddbbeeU, 0x7f60fda3U,  
0x01269ff7U, 0x72f5bc5cU, 0x663bc544U, 0xfb7e345bU,  
0x4329768bU, 0x23c6dccbuU, 0xedfc68b6U, 0xe4f163b8U,  
0x31dccad7U, 0x63851042U, 0x97224013U, 0xc6112084U,  
0x4a247d85U, 0xbb3df8d2U, 0xf93211aeU, 0x29a16dc7U,  
0x9e2f4b1dU, 0xb230f3dcU, 0x8652ec0dU, 0xc1e3d077U,  
0xb3166c2bU, 0x70b999a9U, 0x9448fa11U, 0xe9642247U,  
0xfc8cc4a8U, 0xf03f1aa0U, 0x7d2cd856U, 0x3390ef22U,  
0x494ec787U, 0x38d1c1d9U, 0xcaa2fe8cU, 0xd40b3698U,  
0xf581cfa6U, 0x7ade28a5U, 0xb78e26daU, 0xadbf4a3fU,  
0x3a9de42cU, 0x78920d50U, 0x5fcc9b6aU, 0x7e466254U,  
0x8d13c2f6U, 0xd8b8e890U, 0x39f75e2eU, 0xc3aff582U,  
0x5d80be9fU, 0xd0937c69U, 0xd52da96fU, 0x2512b3cfU,  
0xac993bc8U, 0x187da710U, 0x9c636ee8U, 0x3bbb7bdbU,  
0x267809cdU, 0x5918f46eU, 0x9ab701ecU, 0x4f9aa883U,  
0x956e65e6U, 0xffe67eaaU, 0xbccf0821U, 0x15e8e6efU,  
0xe79bd9baU, 0x6f36ce4aU, 0x9f09d4eaU, 0xb07cd629U,  
0xa4b2af31U, 0x3f23312aU, 0xa59430c6U, 0xa266c035U,  
0x4ebc3774U, 0x82caa6fcU, 0x90d0b0e0U, 0xa7d81533U,  
0x04984af1U, 0xecdaf741U, 0xcd500e7fU, 0x91f62f17U,  
0x4dd68d76U, 0xefb04d43U, 0xaa4d54ccU, 0x9604dfe4U,  
0xd1b5e39eU, 0x6a881b4cU, 0x2c1fb8c1U, 0x65517f46U,  
0x5eea049dU, 0x8c355d01U, 0x877473faU, 0x0b412efbU,  
0x671d5ab3U, 0xdbd25292U, 0x105633e9U, 0xd647136dU,  
0xd7618c9aU, 0xa10c7a37U, 0xf8148e59U, 0x133c89ebU,  
0xa927eeceU, 0x61c935b7U, 0x1ce5ede1U, 0x47b13c7aU,  
0xd2df599cU, 0xf2733f55U, 0x14ce7918U, 0xc737bf73U,  
0xf7cdea53U, 0xfdaa5b5fU, 0x3d6f14dfU, 0x44db8678U,  
0xaff381caU, 0x68c43eb9U, 0x24342c38U, 0xa3405fc2U,  
0x1dc37216U, 0xe2250cbcU, 0x3c498b28U, 0x0d9541ffU,  
0xa8017139U, 0x0cb3de08U, 0xb4e49cd8U, 0x56c19064U,
```

Superseded

```
0xcb84617bU, 0x3b679dU, 0x3e54748U, 0xb574dU,
};
static const u32 t04[256] = {
0x52525252U, 0x09090909U, 0x6a6a6a6aU, 0xd5d5d5d5U,
0x30303030U, 0x36363636U, 0xa5a5a5a5U, 0x38383838U,
0xbfbfbfbfU, 0x40404040U, 0xa3a3a3a3U, 0x9e9e9e9eU,
0x81818181U, 0xf3f3f3f3U, 0xd7d7d7d7U, 0xfbfbfbfbfU,
0x7c7c7c7cU, 0xe3e3e3e3U, 0x39393939U, 0x82828282U,
0x9b9b9b9bU, 0x2f2f2f2fU, 0xffffffffU, 0x87878787U,
0x34343434U, 0x8e8e8e8eU, 0x43434343U, 0x44444444U,
0xc4c4c4c4U, 0xdedededeU, 0xe9e9e9e9U, 0xcbcbcbcbU,
0x54545454U, 0x7b7b7b7bU, 0x94949494U, 0x32323232U,
0xa6a6a6a6U, 0xc2c2c2c2U, 0x23232323U, 0x3d3d3d3dU,
0xeeeeeeeeU, 0x4c4c4c4cU, 0x95959595U, 0x0b0b0b0bU,
0x42424242U, 0xfafafafaU, 0xc3c3c3c3U, 0x4e4e4e4eU,
0x08080808U, 0x2e2e2e2eU, 0xa1a1a1a1U, 0x66666666U,
0x28282828U, 0xd9d9d9d9U, 0x24242424U, 0xb2b2b2b2U,
0x76767676U, 0x5b5b5b5bU, 0xa2a2a2a2U, 0x49494949U,
0x6d6d6d6dU, 0x8b8b8b8bU, 0xd1d1d1d1U, 0x25252525U,
0x72727272U, 0xf8f8f8f8U, 0xf6f6f6f6U, 0x64646464U,
0x86868686U, 0x68686868U, 0x98989898U, 0x16161616U,
0xd4d4d4d4U, 0xa4a4a4a4U, 0x5c5c5c5cU, 0xccccccccU,
0x5d5d5d5dU, 0x65656565U, 0xb6b6b6b6U, 0x92929292U,
0x6c6c6c6cU, 0x70707070U, 0x48484848U, 0x50505050U,
0xfdfdfdfdU, 0xedededU, 0xb9b9b9b9U, 0xdadadadaU,
0x5e5e5e5eU, 0x15151515U, 0x46464646U, 0x57575757U,
0xa7a7a7a7U, 0x8d8d8d8dU, 0x9d9d9d9dU, 0x84848484U,
0x90909090U, 0xd8d8d8d8U, 0xababababU, 0x00000000U,
0x8c8c8c8cU, 0xbcbcbcbcbU, 0xd3d3d3d3U, 0x0a0a0a0aU,
0xf7f7f7f7U, 0xe4e4e4e4U, 0x58585858U, 0x05050505U,
0xb8b8b8b8U, 0xb3b3b3b3U, 0x45454545U, 0x06060606U,
0xd0d0d0d0U, 0x2c2c2c2cU, 0x1e1e1e1eU, 0x8f8f8f8fU,
0xcacacacaU, 0x3f3f3f3fU, 0xf0f0f0f0U, 0x02020202U,
0xc1c1c1c1U, 0xafafafafU, 0xbdbdbdbdbU, 0x03030303U,
0x01010101U, 0x13131313U, 0x8a8a8a8aU, 0x6b6b6b6bU,
0x3a3a3a3aU, 0x91919191U, 0x11111111U, 0x41414141U,
0x4f4f4f4fU, 0x67676767U, 0xdcddcdcdU, 0xeaeaeaeaU,
0x97979797U, 0xf2f2f2f2U, 0xcfcfcfcfU, 0xcecececeU,
0xf0f0f0f0U, 0xb4b4b4b4U, 0xe6e6e6e6U, 0x73737373U,
0x96969696U, 0xacacacacU, 0x74747474U, 0x22222222U,
0xe7e7e7e7U, 0xadadadadU, 0x35353535U, 0x85858585U,
0xe2e2e2e2U, 0xf9f9f9f9U, 0x37373737U, 0xe8e8e8e8U,
0x1c1c1c1cU, 0x75757575U, 0xdfdfdfdfU, 0x6e6e6e6eU,
0x47474747U, 0xf1f1f1f1U, 0x1a1a1a1aU, 0x71717171U,
0x1d1d1d1dU, 0x29292929U, 0xc5c5c5c5U, 0x89898989U,
0x6f6f6f6fU, 0xb7b7b7b7U, 0x62626262U, 0x0e0e0e0eU,
0xaaaaaaaaU, 0x18181818U, 0xbebebebeU, 0x1b1b1b1bU,
0xfcfcfcfcU, 0x56565656U, 0x3e3e3e3eU, 0x4b4b4b4bU,
0xc6c6c6c6U, 0xd2d2d2d2U, 0x79797979U, 0x20202020U,
0x9a9a9a9aU, 0xdbdbdbdbU, 0xc0c0c0c0U, 0xfefefefeU,
0x78787878U, 0xcdcdcdcdU, 0x5a5a5a5aU, 0xf4f4f4f4U,
0x1f1f1f1fU, 0xddddddddU, 0xa8a8a8a8U, 0x33333333U,
0x88888888U, 0x07070707U, 0xc7c7c7c7U, 0x31313131U,
0xb1b1b1b1U, 0x12121212U, 0x10101010U, 0x59595959U,
0x27272727U, 0x80808080U, 0xececececU, 0x5f5f5f5fU,
0x60606060U, 0x51515151U, 0x7f7f7f7fU, 0xa9a9a9a9U,
0x19191919U, 0xb5b5b5b5U, 0x4a4a4a4aU, 0x0d0d0d0dU,
0x2d2d2d2dU, 0xe5e5e5e5U, 0x7a7a7a7aU, 0x9f9f9f9fU,
0x93939393U, 0xc9c9c9c9U, 0x9c9c9c9cU, 0xefefefefU,
0xa0a0a0a0U, 0xe0e0e0e0U, 0x3b3b3b3bU, 0x4d4d4d4dU,
0xaeaeaeaeU, 0x2a2a2a2aU, 0xf5f5f5f5U, 0xb0b0b0b0U,
0xc8c8c8c8U, 0xeb0eb0ebU, 0xbbbb0bbbU, 0x3c3c3c3cU,
0x83838383U, 0x53535353U, 0x99999999U, 0x61616161U,
0x17171717U, 0x2b2b2b2bU, 0x04040404U, 0x7e7e7e7eU,
0xbababababU, 0x77777777U, 0xd6d6d6d6U, 0x26262626U,
0xe1e1e1e1U, 0x69696969U, 0x14141414U, 0x63636363U,
0x55555555U, 0x21212121U, 0x0c0c0c0cU, 0x7d7d7d7dU,
```

Superseded

```

};
static const u32 rcon[] =
    0x01000000, 0x02000000, 0x04000000, 0x08000000,
    0x10000000, 0x20000000, 0x40000000, 0x80000000,
    0x1B000000, 0x36000000, /* for 128-bit blocks, Rijndael never uses more than 10
rcon values */
};

#define SWAP(x) (_lrotl(x, 8) & 0x00ff00ff | _lrotr(x, 8) & 0xff00ff00)

#ifdef _MSC_VER
#define GETU32(p) SWAP(*(u32*)(p))
#define PUTU32(ct, st) { *((u32*)(ct)) = SWAP((st)); }
#else
#define GETU32(pt) (((u32)(pt)[0] << 24) ^ ((u32)(pt)[1] << 16) ^ ((u32)(pt)[2] << 8)
^ ((u32)(pt)[3]))
#define PUTU32(ct, st) { (ct)[0] = (u8)((st) >> 24); (ct)[1] = (u8)((st) >> 16);
(ct)[2] = (u8)((st) >> 8); (ct)[3] = (u8)(st); }
#endif

/**
 * Expand the cipher key into the encryption key schedule.
 *
 * @return the number of rounds for the given cipher key size.
 */

int rijndaelKeySetupEnc(u32 rk[/*4*(Nr + 1)*/], const u8 cipherKey[], int keyBits) {
    int i = 0;
    u32 temp;

    rk[0] = GETU32(cipherKey);
    rk[1] = GETU32(cipherKey + 4);
    rk[2] = GETU32(cipherKey + 8);
    rk[3] = GETU32(cipherKey + 12);
    if (keyBits == 128) {
        for (;;) {
            temp = rk[3];
            rk[4] = rk[0] ^
                (Te4[(temp >> 16) & 0xff] & 0xff000000) ^
                (Te4[(temp >> 8) & 0xff] & 0x00ff0000) ^
                (Te4[(temp) & 0xff] & 0x0000ff00) ^
                (Te4[(temp >> 24)] & 0x000000ff) ^
                rcon[i];
            rk[5] = rk[1] ^ rk[4];
            rk[6] = rk[2] ^ rk[5];
            rk[7] = rk[3] ^ rk[6];
            if (++i == 10) {
                return 10;
            }
            rk += 4;
        }
    }
    rk[4] = GETU32(cipherKey + 16);
    rk[5] = GETU32(cipherKey + 20);
    if (keyBits == 192) {
        for (;;) {
            temp = rk[5];
            rk[6] = rk[0] ^
                (Te4[(temp >> 16) & 0xff] & 0xff000000) ^
                (Te4[(temp >> 8) & 0xff] & 0x00ff0000) ^
                (Te4[(temp) & 0xff] & 0x0000ff00) ^
                (Te4[(temp >> 24)] & 0x000000ff) ^
                rcon[i];
            rk[7] = rk[1] ^ rk[6];
            rk[8] = rk[2] ^ rk[7];
            rk[9] = rk[3] ^ rk[8];
            if (++i == 8) {

```

Superseded

```

    }
    rk[10] = rk[ 4] ^ rk[ 9];
    rk[11] = rk[ 5] ^ rk[10];
    rk += 6;
}

}
rk[6] = GETU32(cipherKey + 24);
rk[7] = GETU32(cipherKey + 28);
if (keyBits == 256) {
    for (;;) {
        temp = rk[ 7];
        rk[ 8] = rk[ 0] ^
            (Te4[(temp >> 16) & 0xff] & 0xff000000) ^
            (Te4[(temp >>  8) & 0xff] & 0x00ff0000) ^
            (Te4[(temp      ) & 0xff] & 0x0000ff00) ^
            (Te4[(temp >> 24)      ] & 0x000000ff) ^
            rcon[i];
        rk[ 9] = rk[ 1] ^ rk[ 8];
        rk[10] = rk[ 2] ^ rk[ 9];
        rk[11] = rk[ 3] ^ rk[10];
        if (++i == 7) {
            return 14;
        }
        temp = rk[11];
        rk[12] = rk[ 4] ^
            (Te4[(temp >> 24)      ] & 0xff000000) ^
            (Te4[(temp >> 16) & 0xff] & 0x00ff0000) ^
            (Te4[(temp >>  8) & 0xff] & 0x0000ff00) ^
            (Te4[(temp      ) & 0xff] & 0x000000ff);
        rk[13] = rk[ 5] ^ rk[12];
        rk[14] = rk[ 6] ^ rk[13];
        rk[15] = rk[ 7] ^ rk[14];

        rk += 8;
    }
}
return 0;
}

/**
 * Expand the cipher key into the decryption key schedule.
 *
 * @return the number of rounds for the given cipher key size.
 */
int rijndaelKeySetupDec(u32 rk[/*4*(Nr + 1)*/], const u8 cipherKey[], int keyBits) {
    int Nr, i, j;
    u32 temp;

    /* expand the cipher key: */
    Nr = rijndaelKeySetupEnc(rk, cipherKey, keyBits);
    /* invert the order of the round keys: */
    for (i = 0, j = 4*Nr; i < j; i += 4, j -= 4) {
        temp = rk[i]; rk[i] = rk[j]; rk[j] = temp;
        temp = rk[i + 1]; rk[i + 1] = rk[j + 1]; rk[j + 1] = temp;
        temp = rk[i + 2]; rk[i + 2] = rk[j + 2]; rk[j + 2] = temp;
        temp = rk[i + 3]; rk[i + 3] = rk[j + 3]; rk[j + 3] = temp;
    }
    /* apply the inverse MixColumn transform to all round keys but the first and the
last: */
    for (i = 1; i < Nr; i++) {
        rk += 4;
        rk[0] =
            Td0[Te4[(rk[0] >> 24)      ] & 0xff] ^
            Td1[Te4[(rk[0] >> 16) & 0xff] & 0xff] ^
            Td2[Te4[(rk[0] >>  8) & 0xff] & 0xff] ^
            Td3[Te4[(rk[0]      ) & 0xff] & 0xff];
    }
}

```

Superseded

```

rk[1] = Td0[Te4[(rk[0] >> 24) & 0xff] ^ rk[0]] ^ rk[0];
rk[1] = Td1[Te4[(rk[1] >> 16) & 0xff] ^ rk[1]] ^ rk[1];
rk[1] = Td2[Te4[(rk[1] >> 8) & 0xff] ^ rk[1]] ^ rk[1];
rk[1] = Td3[Te4[(rk[1] >> 0) & 0xff] ^ rk[1]] ^ rk[1];
rk[2] =
    Td0[Te4[(rk[2] >> 24) & 0xff] ^ rk[2]] ^ rk[2];
rk[2] = Td1[Te4[(rk[2] >> 16) & 0xff] ^ rk[2]] ^ rk[2];
rk[2] = Td2[Te4[(rk[2] >> 8) & 0xff] ^ rk[2]] ^ rk[2];
rk[2] = Td3[Te4[(rk[2] >> 0) & 0xff] ^ rk[2]] ^ rk[2];
rk[3] =
    Td0[Te4[(rk[3] >> 24) & 0xff] ^ rk[3]] ^ rk[3];
rk[3] = Td1[Te4[(rk[3] >> 16) & 0xff] ^ rk[3]] ^ rk[3];
rk[3] = Td2[Te4[(rk[3] >> 8) & 0xff] ^ rk[3]] ^ rk[3];
rk[3] = Td3[Te4[(rk[3] >> 0) & 0xff] ^ rk[3]] ^ rk[3];
}
return Nr;
}

void rijndaelEncrypt(u32 rk[/*4*(Nr + 1)*/], int Nr, const u8 pt[16], u8 ct[16]) {
    u32 s0, s1, s2, s3, t0, t1, t2, t3;
#ifdef FULL_UNROLL
    int r;
#endif /* ?FULL_UNROLL */

    /*
     * map byte array block to cipher state
     * and add initial round key:
     */
    s0 = GETU32(pt) ^ rk[0];
    s1 = GETU32(pt + 4) ^ rk[1];
    s2 = GETU32(pt + 8) ^ rk[2];
    s3 = GETU32(pt + 12) ^ rk[3];
#ifdef FULL_UNROLL
    /* round 1: */
    t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^ Te3[s3 &
0xff] ^ rk[4];
    t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^ Te3[s0 &
0xff] ^ rk[5];
    t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^ Te3[s1 &
0xff] ^ rk[6];
    t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^ Te3[s2 &
0xff] ^ rk[7];
    /* round 2: */
    s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^ Te3[t3 &
0xff] ^ rk[8];
    s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^ Te3[t0 &
0xff] ^ rk[9];
    s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^ Te3[t1 &
0xff] ^ rk[10];
    s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^ Te3[t2 &
0xff] ^ rk[11];
    /* round 3: */
    t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^ Te3[s3 &
0xff] ^ rk[12];
    t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^ Te3[s0 &
0xff] ^ rk[13];
    t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^ Te3[s1 &
0xff] ^ rk[14];
    t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^ Te3[s2 &
0xff] ^ rk[15];
    /* round 4: */
    s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^ Te3[t3 &
0xff] ^ rk[16];
    s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^ Te3[t0 &
0xff] ^ rk[17];

```

Superseded

```

    s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^ Te3[t1 &
0xff] ^ rk[18];
    s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^ Te3[t2 &
0xff] ^ rk[19];
    /* round 5: */
    t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^ Te3[s3 &
0xff] ^ rk[20];
    t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^ Te3[s0 &
0xff] ^ rk[21];
    t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^ Te3[s1 &
0xff] ^ rk[22];
    t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^ Te3[s2 &
0xff] ^ rk[23];
    /* round 6: */
    s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^ Te3[t3 &
0xff] ^ rk[24];
    s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^ Te3[t0 &
0xff] ^ rk[25];
    s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^ Te3[t1 &
0xff] ^ rk[26];
    s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^ Te3[t2 &
0xff] ^ rk[27];
    /* round 7: */
    t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^ Te3[s3 &
0xff] ^ rk[28];
    t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^ Te3[s0 &
0xff] ^ rk[29];
    t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^ Te3[s1 &
0xff] ^ rk[30];
    t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^ Te3[s2 &
0xff] ^ rk[31];
    /* round 8: */
    s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^ Te3[t3 &
0xff] ^ rk[32];
    s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^ Te3[t0 &
0xff] ^ rk[33];
    s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^ Te3[t1 &
0xff] ^ rk[34];
    s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^ Te3[t2 &
0xff] ^ rk[35];
    /* round 9: */
    t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^ Te3[s3 &
0xff] ^ rk[36];
    t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^ Te3[s0 &
0xff] ^ rk[37];
    t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^ Te3[s1 &
0xff] ^ rk[38];
    t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) & 0xff] ^ Te3[s2 &
0xff] ^ rk[39];
    if (Nr > 10) {
        /* round 10: */
        s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) & 0xff] ^
Te3[t3 & 0xff] ^ rk[40];
        s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) & 0xff] ^
Te3[t0 & 0xff] ^ rk[41];
        s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) & 0xff] ^
Te3[t1 & 0xff] ^ rk[42];
        s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^
Te3[t2 & 0xff] ^ rk[43];
        /* round 11: */
        t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) & 0xff] ^
Te3[s3 & 0xff] ^ rk[44];
        t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) & 0xff] ^
Te3[s0 & 0xff] ^ rk[45];
        t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) & 0xff] ^
Te3[s1 & 0xff] ^ rk[46];
    }

```

Superseded

```

t3 = Te0[s3 >> 24] ^ Te1[s >> 16] & 0xff] ^ Te2[(t1 >> 8) & 0xff] ^
Te3[s2 & 0xff] ^ rk[47];
if (Nr > 12) {
    /* round 12: */
    s0 = Te0[t0 >> 24] ^ Te1[(t1 >> 16) & 0xff] ^ Te2[(t2 >> 8) &
0xff] ^ Te3[t3 & 0xff] ^ rk[48];
    s1 = Te0[t1 >> 24] ^ Te1[(t2 >> 16) & 0xff] ^ Te2[(t3 >> 8) &
0xff] ^ Te3[t0 & 0xff] ^ rk[49];
    s2 = Te0[t2 >> 24] ^ Te1[(t3 >> 16) & 0xff] ^ Te2[(t0 >> 8) &
0xff] ^ Te3[t1 & 0xff] ^ rk[50];
    s3 = Te0[t3 >> 24] ^ Te1[(t0 >> 16) & 0xff] ^ Te2[(t1 >> 8) &
0xff] ^ Te3[t2 & 0xff] ^ rk[51];
    /* round 13: */
    t0 = Te0[s0 >> 24] ^ Te1[(s1 >> 16) & 0xff] ^ Te2[(s2 >> 8) &
0xff] ^ Te3[s3 & 0xff] ^ rk[52];
    t1 = Te0[s1 >> 24] ^ Te1[(s2 >> 16) & 0xff] ^ Te2[(s3 >> 8) &
0xff] ^ Te3[s0 & 0xff] ^ rk[53];
    t2 = Te0[s2 >> 24] ^ Te1[(s3 >> 16) & 0xff] ^ Te2[(s0 >> 8) &
0xff] ^ Te3[s1 & 0xff] ^ rk[54];
    t3 = Te0[s3 >> 24] ^ Te1[(s0 >> 16) & 0xff] ^ Te2[(s1 >> 8) &
0xff] ^ Te3[s2 & 0xff] ^ rk[55];
}
rk += Nr << 2;
#else /* !FULL_UNROLL */
/*
 * Nr - 1 full rounds:
 */
r = Nr >> 1;
for (;;) {
    t0 =
        Te0[(s0 >> 24)          ] ^
        Te1[(s1 >> 16) & 0xff] ^
        Te2[(s2 >> 8)  & 0xff] ^
        Te3[(s3          ) & 0xff] ^
        rk[4];

    t1 =
        Te0[(s1 >> 24)          ] ^
        Te1[(s2 >> 16) & 0xff] ^
        Te2[(s3 >> 8)  & 0xff] ^
        Te3[(s0          ) & 0xff] ^
        rk[5];

    t2 =
        Te0[(s2 >> 24)          ] ^
        Te1[(s3 >> 16) & 0xff] ^
        Te2[(s0 >> 8)  & 0xff] ^
        Te3[(s1          ) & 0xff] ^
        rk[6];

    t3 =
        Te0[(s3 >> 24)          ] ^
        Te1[(s0 >> 16) & 0xff] ^
        Te2[(s1 >> 8)  & 0xff] ^
        Te3[(s2          ) & 0xff] ^
        rk[7];

    rk += 8;
    if (--r == 0) {
        break;
    }

    s0 =
        Te0[(t0 >> 24)          ] ^
        Te1[(t1 >> 16) & 0xff] ^
        Te2[(t2 >> 8)  & 0xff] ^
        Te3[(t3          ) & 0xff] ^
        rk[0];

    s1 =

```

Superseded

```

    Te0[(t2 >> 24) ] ^
    Te1[(t3 >> 16) & 0xff] ^
    Te2[(t0 >> 8) & 0xff] ^
    Te3[(t1      ) & 0xff] ^
    rk[1];
    s2 =
        Te0[(t2 >> 24) ] ^
        Te1[(t3 >> 16) & 0xff] ^
        Te2[(t0 >> 8) & 0xff] ^
        Te3[(t1      ) & 0xff] ^
        rk[2];
    s3 =
        Te0[(t3 >> 24) ] ^
        Te1[(t0 >> 16) & 0xff] ^
        Te2[(t1 >> 8) & 0xff] ^
        Te3[(t2      ) & 0xff] ^
        rk[3];
}
#endif /* ?FULL_UNROLL */
/*
 * apply last round and
 * map cipher state to byte array block:
 */
s0 =
    (Te4[(t0 >> 24) ] & 0xff000000) ^
    (Te4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t2 >> 8) & 0xff] & 0x0000ff00) ^
    (Te4[(t3      ) & 0xff] & 0x000000ff) ^
    rk[0];
PUTU32(ct      , s0);
s1 =
    (Te4[(t1 >> 24) ] & 0xff000000) ^
    (Te4[(t2 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t3 >> 8) & 0xff] & 0x0000ff00) ^
    (Te4[(t0      ) & 0xff] & 0x000000ff) ^
    rk[1];
PUTU32(ct + 4, s1);
s2 =
    (Te4[(t2 >> 24) ] & 0xff000000) ^
    (Te4[(t3 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t0 >> 8) & 0xff] & 0x0000ff00) ^
    (Te4[(t1      ) & 0xff] & 0x000000ff) ^
    rk[2];
PUTU32(ct + 8, s2);
s3 =
    (Te4[(t3 >> 24) ] & 0xff000000) ^
    (Te4[(t0 >> 16) & 0xff] & 0x00ff0000) ^
    (Te4[(t1 >> 8) & 0xff] & 0x0000ff00) ^
    (Te4[(t2      ) & 0xff] & 0x000000ff) ^
    rk[3];
PUTU32(ct + 12, s3);
}

void rijndaelDecrypt(u32 rk[/*4*(Nr + 1)*/], int Nr, const u8 ct[16], u8 pt[16]) {
    u32 s0, s1, s2, s3, t0, t1, t2, t3;
#ifdef FULL_UNROLL
    int r;
#endif /* ?FULL_UNROLL */

    /*
     * map byte array block to cipher state
     * and add initial round key:
     */
    s0 = GETU32(ct      ) ^ rk[0];
    s1 = GETU32(ct + 4) ^ rk[1];
    s2 = GETU32(ct + 8) ^ rk[2];
    s3 = GETU32(ct + 12) ^ rk[3];

```

Superseded

```

#ifdef FULL_UNROLL
/* round 1: */
t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) & 0xff] ^ Td3[s1 &
0xff] ^ rk[4];
t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) & 0xff] ^ Td3[s2 &
0xff] ^ rk[5];
t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) & 0xff] ^ Td3[s3 &
0xff] ^ rk[6];
t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) & 0xff] ^ Td3[s0 &
0xff] ^ rk[7];
/* round 2: */
s0 = Td0[t0 >> 24] ^ Td1[(t3 >> 16) & 0xff] ^ Td2[(t2 >> 8) & 0xff] ^ Td3[t1 &
0xff] ^ rk[8];
s1 = Td0[t1 >> 24] ^ Td1[(t0 >> 16) & 0xff] ^ Td2[(t3 >> 8) & 0xff] ^ Td3[t2 &
0xff] ^ rk[9];
s2 = Td0[t2 >> 24] ^ Td1[(t1 >> 16) & 0xff] ^ Td2[(t0 >> 8) & 0xff] ^ Td3[t3 &
0xff] ^ rk[10];
s3 = Td0[t3 >> 24] ^ Td1[(t2 >> 16) & 0xff] ^ Td2[(t1 >> 8) & 0xff] ^ Td3[t0 &
0xff] ^ rk[11];
/* round 3: */
t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) & 0xff] ^ Td3[s1 &
0xff] ^ rk[12];
t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) & 0xff] ^ Td3[s2 &
0xff] ^ rk[13];
t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) & 0xff] ^ Td3[s3 &
0xff] ^ rk[14];
t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) & 0xff] ^ Td3[s0 &
0xff] ^ rk[15];
/* round 4: */
s0 = Td0[t0 >> 24] ^ Td1[(t3 >> 16) & 0xff] ^ Td2[(t2 >> 8) & 0xff] ^ Td3[t1 &
0xff] ^ rk[16];
s1 = Td0[t1 >> 24] ^ Td1[(t0 >> 16) & 0xff] ^ Td2[(t3 >> 8) & 0xff] ^ Td3[t2 &
0xff] ^ rk[17];
s2 = Td0[t2 >> 24] ^ Td1[(t1 >> 16) & 0xff] ^ Td2[(t0 >> 8) & 0xff] ^ Td3[t3 &
0xff] ^ rk[18];
s3 = Td0[t3 >> 24] ^ Td1[(t2 >> 16) & 0xff] ^ Td2[(t1 >> 8) & 0xff] ^ Td3[t0 &
0xff] ^ rk[19];
/* round 5: */
t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) & 0xff] ^ Td3[s1 &
0xff] ^ rk[20];
t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) & 0xff] ^ Td3[s2 &
0xff] ^ rk[21];
t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) & 0xff] ^ Td3[s3 &
0xff] ^ rk[22];
t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) & 0xff] ^ Td3[s0 &
0xff] ^ rk[23];
/* round 6: */
s0 = Td0[t0 >> 24] ^ Td1[(t3 >> 16) & 0xff] ^ Td2[(t2 >> 8) & 0xff] ^ Td3[t1 &
0xff] ^ rk[24];
s1 = Td0[t1 >> 24] ^ Td1[(t0 >> 16) & 0xff] ^ Td2[(t3 >> 8) & 0xff] ^ Td3[t2 &
0xff] ^ rk[25];
s2 = Td0[t2 >> 24] ^ Td1[(t1 >> 16) & 0xff] ^ Td2[(t0 >> 8) & 0xff] ^ Td3[t3 &
0xff] ^ rk[26];
s3 = Td0[t3 >> 24] ^ Td1[(t2 >> 16) & 0xff] ^ Td2[(t1 >> 8) & 0xff] ^ Td3[t0 &
0xff] ^ rk[27];
/* round 7: */
t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) & 0xff] ^ Td3[s1 &
0xff] ^ rk[28];
t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) & 0xff] ^ Td3[s2 &
0xff] ^ rk[29];
t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) & 0xff] ^ Td3[s3 &
0xff] ^ rk[30];
t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) & 0xff] ^ Td3[s0 &
0xff] ^ rk[31];
/* round 8: */
s0 = Td0[t0 >> 24] ^ Td1[(t3 >> 16) & 0xff] ^ Td2[(t2 >> 8) & 0xff] ^ Td3[t1 &
0xff] ^ rk[32];

```

Superseded

```

    s1 = Td0[t1 >> 24] ^ Td1[(t0 >> 16) & 0xff] ^ Td2[(t3 >> 8) & 0xff] ^ Td3[t2 &
0xff] ^ rk[33];
    s2 = Td0[t2 >> 24] ^ Td1[(t1 >> 16) & 0xff] ^ Td2[(t0 >> 8) & 0xff] ^ Td3[t3 &
0xff] ^ rk[34];
    s3 = Td0[t3 >> 24] ^ Td1[(t2 >> 16) & 0xff] ^ Td2[(t1 >> 8) & 0xff] ^ Td3[t0 &
0xff] ^ rk[35];
    /* round 9: */
    t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) & 0xff] ^ Td3[s1 &
0xff] ^ rk[36];
    t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) & 0xff] ^ Td3[s2 &
0xff] ^ rk[37];
    t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) & 0xff] ^ Td3[s3 &
0xff] ^ rk[38];
    t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) & 0xff] ^ Td3[s0 &
0xff] ^ rk[39];
    if (Nr > 10) {
        /* round 10: */
        s0 = Td0[t0 >> 24] ^ Td1[(t3 >> 16) & 0xff] ^ Td2[(t2 >> 8) & 0xff] ^
Td3[t1 & 0xff] ^ rk[40];
        s1 = Td0[t1 >> 24] ^ Td1[(t0 >> 16) & 0xff] ^ Td2[(t3 >> 8) & 0xff] ^
Td3[t2 & 0xff] ^ rk[41];
        s2 = Td0[t2 >> 24] ^ Td1[(t1 >> 16) & 0xff] ^ Td2[(t0 >> 8) & 0xff] ^
Td3[t3 & 0xff] ^ rk[42];
        s3 = Td0[t3 >> 24] ^ Td1[(t2 >> 16) & 0xff] ^ Td2[(t1 >> 8) & 0xff] ^
Td3[t0 & 0xff] ^ rk[43];
        /* round 11: */
        t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) & 0xff] ^
Td3[s1 & 0xff] ^ rk[44];
        t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) & 0xff] ^
Td3[s2 & 0xff] ^ rk[45];
        t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) & 0xff] ^
Td3[s3 & 0xff] ^ rk[46];
        t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) & 0xff] ^
Td3[s0 & 0xff] ^ rk[47];
        if (Nr > 12) {
            /* round 12: */
            s0 = Td0[t0 >> 24] ^ Td1[(t3 >> 16) & 0xff] ^ Td2[(t2 >> 8) &
0xff] ^ Td3[t1 & 0xff] ^ rk[48];
            s1 = Td0[t1 >> 24] ^ Td1[(t0 >> 16) & 0xff] ^ Td2[(t3 >> 8) &
0xff] ^ Td3[t2 & 0xff] ^ rk[49];
            s2 = Td0[t2 >> 24] ^ Td1[(t1 >> 16) & 0xff] ^ Td2[(t0 >> 8) &
0xff] ^ Td3[t3 & 0xff] ^ rk[50];
            s3 = Td0[t3 >> 24] ^ Td1[(t2 >> 16) & 0xff] ^ Td2[(t1 >> 8) &
0xff] ^ Td3[t0 & 0xff] ^ rk[51];
            /* round 13: */
            t0 = Td0[s0 >> 24] ^ Td1[(s3 >> 16) & 0xff] ^ Td2[(s2 >> 8) &
0xff] ^ Td3[s1 & 0xff] ^ rk[52];
            t1 = Td0[s1 >> 24] ^ Td1[(s0 >> 16) & 0xff] ^ Td2[(s3 >> 8) &
0xff] ^ Td3[s2 & 0xff] ^ rk[53];
            t2 = Td0[s2 >> 24] ^ Td1[(s1 >> 16) & 0xff] ^ Td2[(s0 >> 8) &
0xff] ^ Td3[s3 & 0xff] ^ rk[54];
            t3 = Td0[s3 >> 24] ^ Td1[(s2 >> 16) & 0xff] ^ Td2[(s1 >> 8) &
0xff] ^ Td3[s0 & 0xff] ^ rk[55];
        }
    }
    rk += Nr << 2;
#else
    /* !FULL_UNROLL */
    /*
     * Nr - 1 full rounds:
     */
    r = Nr >> 1;
    for (;;) {
        t0 =
            Td0[(s0 >> 24)
            Td1[(s3 >> 16) & 0xff]
            Td2[(s2 >> 8) & 0xff]
            Td3[(s1
                ) & 0xff]

```

Superseded

```

rk[4];
t1 =
    Td0[(s1 >> 24) ] ^
    Td1[(s0 >> 16) & 0xff] ^
    Td2[(s3 >> 8) & 0xff] ^
    Td3[(s2      ) & 0xff] ^
    rk[5];
t2 =
    Td0[(s2 >> 24) ] ^
    Td1[(s1 >> 16) & 0xff] ^
    Td2[(s0 >> 8) & 0xff] ^
    Td3[(s3      ) & 0xff] ^
    rk[6];
t3 =
    Td0[(s3 >> 24) ] ^
    Td1[(s2 >> 16) & 0xff] ^
    Td2[(s1 >> 8) & 0xff] ^
    Td3[(s0      ) & 0xff] ^
    rk[7];

rk += 8;
if (--r == 0) {
    break;
}

s0 =
    Td0[(t0 >> 24) ] ^
    Td1[(t3 >> 16) & 0xff] ^
    Td2[(t2 >> 8) & 0xff] ^
    Td3[(t1      ) & 0xff] ^
    rk[0];
s1 =
    Td0[(t1 >> 24) ] ^
    Td1[(t0 >> 16) & 0xff] ^
    Td2[(t3 >> 8) & 0xff] ^
    Td3[(t2      ) & 0xff] ^
    rk[1];
s2 =
    Td0[(t2 >> 24) ] ^
    Td1[(t1 >> 16) & 0xff] ^
    Td2[(t0 >> 8) & 0xff] ^
    Td3[(t3      ) & 0xff] ^
    rk[2];
s3 =
    Td0[(t3 >> 24) ] ^
    Td1[(t2 >> 16) & 0xff] ^
    Td2[(t1 >> 8) & 0xff] ^
    Td3[(t0      ) & 0xff] ^
    rk[3];
}
#endif /* ?FULL_UNROLL */
/*
 * apply last round and
 * map cipher state to byte array block:
 */
s0 =
    (Td4[(t0 >> 24) ] & 0xff000000) ^
    (Td4[(t3 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(t2 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(t1      ) & 0xff] & 0x000000ff) ^
    rk[0];
PUTU32(pt, s0);
s1 =
    (Td4[(t1 >> 24) ] & 0xff000000) ^
    (Td4[(t0 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(t3 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(t2      ) & 0xff] & 0x000000ff) ^

```

Superseded

```

        rk[1],
PUTU32(pt + 4, 1)
s2 =
    (Td4[(t2 >> 24)          ] & 0xff000000) ^
    (Td4[(t1 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(t0 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(t3          ) & 0xff] & 0x000000ff) ^
    rk[2];
PUTU32(pt + 8, s2);
s3 =
    (Td4[(t3 >> 24)          ] & 0xff000000) ^
    (Td4[(t2 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(t1 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(t0          ) & 0xff] & 0x000000ff) ^
    rk[3];
PUTU32(pt + 12, s3);
}

#ifdef INTERMEDIATE_VALUE_KAT

void rijndaelEncryptRound(const u32 rk[/*4*(Nr + 1)*/], int Nr, u8 block[16], int
rounds) {
    int r;
    u32 s0, s1, s2, s3, t0, t1, t2, t3;

    /*
     * map byte array block to cipher state
     * and add initial round key:
     */
    s0 = GETU32(block          ) ^ rk[0];
    s1 = GETU32(block + 4) ^ rk[1];
    s2 = GETU32(block + 8) ^ rk[2];
    s3 = GETU32(block + 12) ^ rk[3];
    rk += 4;

    /*
     * Nr - 1 full rounds:
     */
    for (r = (rounds < Nr ? rounds : Nr - 1); r > 0; r--) {
        t0 =
            Te0[(s0 >> 24)          ] ^
            Te1[(s1 >> 16) & 0xff] ^
            Te2[(s2 >> 8) & 0xff] ^
            Te3[(s3          ) & 0xff] ^
            rk[0];

        t1 =
            Te0[(s1 >> 24)          ] ^
            Te1[(s2 >> 16) & 0xff] ^
            Te2[(s3 >> 8) & 0xff] ^
            Te3[(s0          ) & 0xff] ^
            rk[1];

        t2 =
            Te0[(s2 >> 24)          ] ^
            Te1[(s3 >> 16) & 0xff] ^
            Te2[(s0 >> 8) & 0xff] ^
            Te3[(s1          ) & 0xff] ^
            rk[2];

        t3 =
            Te0[(s3 >> 24)          ] ^
            Te1[(s0 >> 16) & 0xff] ^
            Te2[(s1 >> 8) & 0xff] ^
            Te3[(s2          ) & 0xff] ^
            rk[3];

        s0 = t0;
        s1 = t1;
        s2 = t2;
    }
}

```

Superseded

```

    s3 = t0;
    rk += 4;
}

/*
 * apply last round and
 * map cipher state to byte array block:
 */
if (rounds == Nr) {
    t0 =
        (Te4[(s0 >> 24)          ] & 0xff000000) ^
        (Te4[(s1 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(s2 >> 8) & 0xff] & 0x0000ff00) ^
        (Te4[(s3          ) & 0xff] & 0x000000ff) ^
        rk[0];

    t1 =
        (Te4[(s1 >> 24)          ] & 0xff000000) ^
        (Te4[(s2 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(s3 >> 8) & 0xff] & 0x0000ff00) ^
        (Te4[(s0          ) & 0xff] & 0x000000ff) ^
        rk[1];

    t2 =
        (Te4[(s2 >> 24)          ] & 0xff000000) ^
        (Te4[(s3 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(s0 >> 8) & 0xff] & 0x0000ff00) ^
        (Te4[(s1          ) & 0xff] & 0x000000ff) ^
        rk[2];

    t3 =
        (Te4[(s3 >> 24)          ] & 0xff000000) ^
        (Te4[(s0 >> 16) & 0xff] & 0x00ff0000) ^
        (Te4[(s1 >> 8) & 0xff] & 0x0000ff00) ^
        (Te4[(s2          ) & 0xff] & 0x000000ff) ^
        rk[3];

    s0 = t0;
    s1 = t1;
    s2 = t2;
    s3 = t3;
}

PUTU32(block      , s0);
PUTU32(block + 4, s1);
PUTU32(block + 8, s2);
PUTU32(block + 12, s3);
}

void rijndaelDecryptRound(const u32 rk[/*4*(Nr + 1)*/], int Nr, u8 block[16], int
rounds) {
    int r;
    u32 s0, s1, s2, s3, t0, t1, t2, t3;

    /*
     * map byte array block to cipher state
     * and add initial round key:
     */
    s0 = GETU32(block      ) ^ rk[0];
    s1 = GETU32(block + 4) ^ rk[1];
    s2 = GETU32(block + 8) ^ rk[2];
    s3 = GETU32(block + 12) ^ rk[3];
    rk += 4;

    /*
     * Nr - 1 full rounds:
     */
    for (r = (rounds < Nr ? rounds : Nr) - 1; r > 0; r--) {
        t0 =

```

Superseded

```

    Td0[(s0 >> 24) ] ^
    Td1[(s0 >> 16) & 0xff] ^
    Td2[(s2 >> 8) & 0xff] ^
    Td3[(s1      ) & 0xff] ^
    rk[0];

t1 =
    Td0[(s1 >> 24) ] ^
    Td1[(s0 >> 16) & 0xff] ^
    Td2[(s3 >> 8) & 0xff] ^
    Td3[(s2      ) & 0xff] ^
    rk[1];

t2 =
    Td0[(s2 >> 24) ] ^
    Td1[(s1 >> 16) & 0xff] ^
    Td2[(s0 >> 8) & 0xff] ^
    Td3[(s3      ) & 0xff] ^
    rk[2];

t3 =
    Td0[(s3 >> 24) ] ^
    Td1[(s2 >> 16) & 0xff] ^
    Td2[(s1 >> 8) & 0xff] ^
    Td3[(s0      ) & 0xff] ^
    rk[3];

s0 = t0;
s1 = t1;
s2 = t2;
s3 = t3;
rk += 4;

}

/*
 * complete the last round and
 * map cipher state to byte array block:
 */
t0 =
    (Td4[(s0 >> 24) ] & 0xff000000) ^
    (Td4[(s3 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(s2 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(s1      ) & 0xff] & 0x000000ff);

t1 =
    (Td4[(s1 >> 24) ] & 0xff000000) ^
    (Td4[(s0 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(s3 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(s2      ) & 0xff] & 0x000000ff);

t2 =
    (Td4[(s2 >> 24) ] & 0xff000000) ^
    (Td4[(s1 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(s0 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(s3      ) & 0xff] & 0x000000ff);

t3 =
    (Td4[(s3 >> 24) ] & 0xff000000) ^
    (Td4[(s2 >> 16) & 0xff] & 0x00ff0000) ^
    (Td4[(s1 >> 8) & 0xff] & 0x0000ff00) ^
    (Td4[(s0      ) & 0xff] & 0x000000ff);

if (rounds == Nr) {
    t0 ^= rk[0];
    t1 ^= rk[1];
    t2 ^= rk[2];
    t3 ^= rk[3];
}

PUTU32(block      , t0);
PUTU32(block + 4, t1);
PUTU32(block + 8, t2);

```

Superseded

```

    PUTU32(block_1, t3)
}
#endif /* INTERMEDIATE_VALUE_KAT */

static void block_init(block_state *state, unsigned char *key,
                      int keylen)
{
    int Nr = 0;

    if (keylen != 16 && keylen != 24 && keylen != 32) {
        PyErr_SetString(PyExc_ValueError,
                        "AES key must be either 16, 24, or 32 bytes long");
        return;
    }
    switch (keylen) {
        case(16): Nr = 10; break;
        case(24): Nr = 12; break;
        case(32): Nr = 14; break;
    }
    state->rounds = Nr;
    rijndaelKeySetupEnc(state->ek, key, keylen*8);
    rijndaelKeySetupDec(state->dk, key, keylen*8);
}

static void block_encrypt(block_state *self, u8 *in, u8 *out)
{
    rijndaelEncrypt(self->ek, self->rounds, in, out);
}

static void block_decrypt(block_state *self, u8 *in, u8 *out)
{
    rijndaelDecrypt(self->dk, self->rounds, in, out);
}

```

Superseded

Appendix III Acknowledgments

On behalf of our industry, we would like to thank the following individuals for their contributions to the development of this specification.

Contributor	Company Affiliation
John Dickinson	Bright House Networks
Andrew Chagnon, Drew Davis, James Fletcher, Paul Gray	Broadcom
Mike Holmes	Broadway Networks
Curtis Knittle, Bob Lund, Glenn Russell	CableLabs
Shamim Akhtar, Doug Jones, Mehmet Toy	Comcast
Vladimir Bronstein, James Chen, Hesham ElBakoury	Hitachi Communication Technologies America
Victor Blake	Independent Consultant
Robert Harris, Karen Rice, Ashish Sardesai	Time Warner Cable
Marek Hajduczenia, Nevin Jones, Zang Meiyen	ZTE

In addition to the above, we would like to thank the following individuals who provided subject matter expertise on one or more aspects of the project:

Contributor	Company Affiliation
Edwin Mallette	Bright House Networks
Andrew Dellow, Niki Pantelias, Ricki Li, Paul Runcy, Ed Boyd, Howard Abramson, Matt Hartling	Broadcom
Jianhui Zhou, Wen Li, Fulin Pan	Broadway Networks
Brian Hedstrom, Stuart Hoggan, Greg White, Karthik Sundaresan, Chris Donley	CableLabs
Jason Combs, Saif Rahman, Matt Scully, Bin Wen, Philip Chang, Rashid Siddiqui	Comcast
Dylan Ko, Simon Zhu, Guru Yeleswarapu, Jeff Stribling	Hitachi
Ron daSilva, Mike Kelsen, Shan Huang, Matt Cannon, Tushar Nakhre	Time Warner Cable
Stove Li Zhang, David Chen, Dick Chen	ZTE