Telecommunications and Information Exchange Between Systems

ISO/IEC JTC 1/SC 6

Document Number:	N13774
Date:	2008-11-18
Replaces:	
Document Type:	Other document (Defined)
Document Title:	WAPI Independent IS Sample Draft
Document Source:	SC 6/WG 1 Montreux meeting
Project Number:	
Document Status:	As per the SC 6 Montreux resolution 6.1.5, this document is circulated to SC 6 national bodies and IEEE liaison for three month review and comment.
Action ID:	COM
Due Date:	2009-02-18
No. of Pages:	154

ISO/IEC JTC1/SC6 Secretariat Ms. Jooran Lee, KSA (on behalf of KATS)

Korea Technology Center #701-7 Yeoksam-dong, Gangnam-gu, Seoul, 135-513, Republic of Korea ;

Telephone: +82 2 6009 4808 ; Facsimile: +82 2 6009 4819 ; Email : jooran@kisi.or.kr

INTERNATIONAL STANDARD **ISO/IEC 8802-XX**

(or other number, to be determined by ITTF)

XXXXX edition
CCYY-MM-DD

Information technology— Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements —

Part XX:

Alternative Security Mechanism to be used with ISO/IEC 8802-11

1. Overview

1.1 Scope

The scope of this standard is to develop a medium access control (MAC) and physical layer (PHY) specification for wireless connectivity for fixed, portable, and moving stations within a local area. Specially, in this international standard an enhanced security specification — WLAN Authentication and Privacy Infrastructure (WAPI) is defined.

1.2 Purpose

The purpose of this standard is to provide wireless connectivity to automatic machinery, equipment, or stations that require rapid deployment, which may be portable or hand-held, or which may be mounted on moving vehicles within a local area. This standard also offers regulatory bodies a means of standardizing access to one or more frequency bands for the purpose of local area communication. For the coexistence of different security mechanisms, this document mainly provides an enhanced security mechanism WAPI, which is different from those of ISO/IEC 8802-11: 2006.

Note: “ISO/IEC 8802-11: 2006” is a combination of four following documents: ISO/IEC 8802-11:2005 Information technology — Telecommunications and information exchange between systems — Local and metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, and its AMENDMENT 4: Further Higher Data Rate Extension in the 2.4 GHz Band, AMENDMENT 5: Spectrum and Transmit Power Management Extensions in the 5 GHz band in Europe, and AMENDMENT 6: Medium Access Control (MAC) Security Enhancements.

2. Normative references

The following standards contain provisions which, through references in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

IEEE Std 802-1990, IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture.

IEEE Std C95.1-1991 (Reaff 1997), IEEE Standard Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300 GHz.

ISO/IEC 7498-1: 1994, Information technology—Open Systems Interconnection—Basic Reference Model: The Basic Model.

ISO/IEC 8802-2: 1998, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 2: Logical link control.

ISO/IEC 8802-11:2005 Information technology—Telecommunications and information exchange between

systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

ISO/IEC 8802-11:2005 Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, AMENDMENT 4: Further Higher Data Rate Extension in the 2.4 GHz Band.

ISO/IEC 8802-11:2005 Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, AMENDMENT 5: Spectrum and Transmit Power Management Extensions in the 5 GHz band in Europe.

ISO/IEC 8802-11:2005 Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, AMENDMENT 6: Medium Access Control (MAC) Security Enhancements.

ISO/IEC 8824-1: 1995, Information technology—Abstract Syntax Notation One (ASN.1): Specification of basic notation.

ISO/IEC 8824-2: 1995, Information technology—Abstract Syntax Notation One (ASN.1): Information object specification.

ISO/IEC 8824-3: 1995, Information technology—Abstract Syntax Notation One (ASN.1): Constraint specification.

ISO/IEC 8824-4: 1995, Information technology—Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.

ISO/IEC 8825-1: 1995, Information technology—ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).

ISO/IEC 8825-2: 1996, Information technology—ASN.1 encoding rules: Specification of Packed Encoding Rules (PER).

ISO/IEC 15802-1: 1995, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 1: Medium Access Control (MAC) service definition.

ITU Radio Regulations, volumes 1–4.

ITU-T Recommendation 210 (11/93), Information technology—Open systems interconnection—Basic Reference Model: Conventions for the definition of OSI services (*common text with ISO/IEC*).

ITU-T Recommendation Z.100 (03/93), CCITT specification and description language (SDL).

ITU-T Recommendation Z.105 (03/95), SDL combined with ASN.1 (SDL/ASN.1).

ANSI X9.62, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA).

FIPS 180-2, Secure Hash Standard (SHS), August 2002.

IETF RFC2104, HMAC: Keyed-Hashing for Message Authentication.

ITU-T Recommendation X.509, Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks.

ISO/IEC 15946, Information technology - Security techniques - Cryptographic techniques based on elliptic curves.

ISO/IEC 10116: 1997 (2nd edition), Information technology - Security techniques - Modes of operation for an n-bit block cipher algorithm.

3. Definitions

For the purposes of this standard, the following terms and definitions apply.

3.1 access control: The prevention of unauthorized usage of resources.

3.2 access point (AP): Any entity that has station functionality and provides access to the distribution services, via the wireless medium (WM) for associated stations.

3.3 association: The service used to establish access point/station (AP/STA) mapping and enable STA invocation of the distribution system services (DSSs).

3.4 authentication: The service used to establish the identity of one station as a member of the set of stations authorized to associate with another station.

3.5 deauthentication: The service that voids an existing authentication relationship.

3.6 disassociation: The service that removes an existing association.

3.7 independent basic service set (IBSS): A BSS that forms a self-contained network, and in which no access to a distribution system (DS) is available.

3.8 infrastructure: The infrastructure includes the distribution system medium (DSM), access point (AP), and

portal entities. It is also the logical location of distribution and integration service functions of an extended service set (ESS). An infrastructure contains one or more APs and zero or more portals in addition to the distribution system (DS).

3.9 medium access control (MAC) management protocol data unit (MMPDU): The unit of data exchanged between two peer MAC entities to implement the MAC management protocol.

3.10 medium access control (MAC) protocol data unit (MPDU): The unit of data exchanged between two peer MAC entities using the services of the physical layer (PHY).

3.11 medium access control (MAC) service data unit (MSDU): Information that is delivered as a unit between MAC service access points (SAPs).

3.12 multicast: A medium access control (MAC) address that has the group bit set. A multicast MAC service data unit (MSDU) is one with a multicast destination address. A multicast MAC protocol data unit (MPDU) or control frame is one with a multicast receiver address.

3.13 Confidentiality: The service used to prevent the content of messages from being read by other than the intended recipients.

3.14 reassociation: The service that enables an established association [between access point (AP) and station (STA)] to be transferred from one AP to another (or the same) AP.

3.15 station (STA): Any device that contains an ISO/IEC 8802-11 conformant medium access control (MAC) and physical layer (PHY) interface to the wireless medium (WM).

3.16 unauthorized disclosure: The process of making information available to unauthorized individuals, entities, or processes.

3.17 unauthorized resource use: Use of a resource not consistent with the defined security policy.

3.18 unicast frame: A frame that is addressed to a single recipient, not a broadcast or multicast frame. *Syn:* directed address.

3.19 wireless local area network authentication and privacy infrastructure (WAPI): A security scheme that is stated to provide identity authentication and data confidentiality for WLAN. WAPI comprises WAI (WLAN Authentication Infrastructure) and WPI (WLAN Privacy Infrastructure).

3.20 wireless medium (WM): The medium used to implement the transfer of protocol data units (PDUs) between peer physical layer (PHY) entities of a wireless local area network (LAN).

3.21 authenticator entity (AE): An entity that provides authentication action for the authentication supplicant before the supplicant accesses the network. It resides in a STA or an AP.

3.22 authentication and key management (AKM) suite: A set of methods for authentication and key management.

3.23 authentication service entity (ASE): An entity that provides mutual identity authentication for the AE and the ASUE. It resides in any ASU.

3.24 authentication service unit (ASU): An important part of the WAI authentication framework that is based on a public-key cryptography. It manages the certificates and authenticates the identities of users.

3.25 authentication supplicant entity (ASUE): An entity that requires an identity authentication through the ASU. It resides in any STA.

3.26 Big-endian: The concept that, for a given multi-octet numeric representation, the most significant octet has the lowest address.

3.27 base key (BK): An auxiliary key used to generate the unicast session key. It can be negotiated in the certificate authentication process or directly derived from the PSK.

3.28 base key security association (BKSA): The result of the certificate authentication process or the result directly derived from the PSK.

3.29 key encryption key (KEK): A key used to encrypt the key data field in the key management protocol.

3.30 Little-endian: The concept that, for a given multi-octet numeric representation, the least significant octet has the lowest address.

3.31 message authentication key (MAK): A key used to authenticate the data source and check integrity in the key management protocol.

3.32 message integrity code (MIC): A value generated by using a symmetric key cryptographic algorithm on the input data. If the input data are changed, a new value cannot be correctly computed without knowledge of the symmetric key.

3.33 multicast session key (MSK): A random value used to protect multicast MPDUs sent by the node. It is derived from the multicast master key.

3.34 multicast session key security association (MSKSA): The result of the multicast key announcement process.

3.35 notification master key(NMK): An auxiliary key used to generate the multicast encryption key and the multicast integrity check key. NMK is used as multicast master key in multicast key announcement process. NMK is used as STA-to-STA master key in STA-to-STA unicast key negotiation process.

3.36 preshared key (PSK): A static key distributed to the STA. The way of the key distribution is beyond this

specification.

3.37 STAKey: A symmetric key used to protect direct communications between two stations in a BSS.

3.38 STAKey security association (STAKeySA): The result of the STA-to-STA unicast key negotiation process in a BSS. A STAKeySA includes a STAKey.

3.39 unicast session key (USK): A random value that is derived from the base key with a pseudo-random function. It is composed of four parts: the unicast encryption key (UEK), the unicast integrity check key (UCK), the message authentication key (MAK) and the key encryption key (KEK).

3.40 unicast session key security association (USKSA): The result of the unicast key negotiation process.

3.41 WLAN Authentication Infrastructure (WAI): A security scheme defined in this specification which is used for identity authentication and key management in WLAN.

3.42 WAPI key management: The key management including the unicast key negotiation, the multicast key announcement and the STAKey announcement.

3.43 WAPI security network: A network with the WAPI security mechanism. It is identified by the WAPI parameter set in management frames such as the Beacon frame.

3.44 wired equivalent privacy (WEP): The optional cryptographic confidentiality algorithm specified by IEEE 802.11 used to provide data confidentiality that is subjectively equivalent to the confidentiality of a wired local area network (LAN) medium that does not employ cryptographic techniques to enhance privacy.

3.45 station service (SS): The set of services that support transport of medium access control (MAC) ser-vice data units (MSDUs) between stations within a basic service set (BSS)

3.46 network allocation vector (NAV): An indicator, maintained by each station, of time periods when transmission onto the wireless medium (WM) will not be initiated by the station whether or not the station's clear channel assessment (CCA) function senses that the WM is busy.

3.47 extended service set (ESS): A set of one or more interconnected basic service sets (BSSs) and integrated local area networks (LANs) that appears as a single BSS to the logical link control layer at any station associated with one of those BSSs.

3.48 distribution: The service that, by using association information, delivers medium access control (MAC) service data units (MSDUs) within the distribution system (DS).

3.49 distribution system (DS): A system used to interconnect a set of basic service sets (BSSs) and integrated local area networks (LANs) to create an extended service set (ESS).

3.50 distribution system medium (DSM): The medium or set of media used by a distribution system (DS) for

communications between access points (APs) and portals of an extended service set (ESS).

3.51distribution system service (DSS): The set of services provided by the distribution system (DS) that enable the medium access control (MAC) to transport MAC service data units (MSDUs) between stations that are not in direct communication with each other over a single instance of the wireless medium (WM). These services include transport of MSDUs between the access points (APs) of basic service sets (BSSs) within an extended service set (ESS), transport of MSDUs between portals and BSSs within an ESS, and transport of MSDUs between stations in the same BSS in cases where the MSDU has a multicast or broad-cast destination address or where the destination is an individual address, but the station sending the MSDU chooses to involve DSS. DSSs are provided between pairs of ISO/IEC 8802-11 MACs.

3.52 basic service set (BSS): A set of stations controlled by a single coordination function.

3.53coordination function: The logical function that determines when a station operating within a basic service set (BSS) is permitted to transmit and may be able to receive protocol data units (PDUs) via the wire-less medium (WM). The coordination function within a BSS may have one point coordination function (PCF) and will have one distributed coordination function (DCF).

3.54 coordination function pollable: A station able to (1) respond to a coordination function poll with a data frame, if such a frame is queued and able to be generated, and (2) interpret acknowledgments in frames sent to or from the point coordinator.

4. Abbreviations and acronyms

ACK	acknowledgment
ADDID	address index
AE	authentication entity
AKM	authentication and key management
AKMP	authentication and key management protocol
AP	access point
ASE	authentication service entity
ASU	authentication service unit
ASUE	authentication supplicant entity
BK	base key
BKID	base key identification
BKSA	base key security association
BSS	basic service set
BSSID	basic service set identification
CRC	cyclic redundancy code
CFP	contention-free period
DA	destination address
DS	distribution system
DSM	distribution system medium

DSS	distribution system service
ECC	ellipse curve cipher
ECDH	Diffie-Hellman exchange in elliptic curve cryptosystem
ESS	extended service set
FCS	frame check sequence
FH	frequency hopping
IBSS	independent basic service set
KEK	key encryption key
LAN	local area network
LLC	logical link control
LME	layer management entity
MAC	medium access control
MAK	message authentication key
MCK	multicast integrity check key
MEK	multicast key encryption key
MIB	management information base
MIC	message integrity check code
MLME	MAC sublayer management entity
MMPDU	MAC management protocol data unit
MPDU	MAC protocol data unit
MSDU	MAC service data unit
MSK	multicast session key
MSKID	multicast session key index
MSKSA	multicast session key security association
NAV	network allocation vector
NMK	notification master key
OID	object identifier
OSI	open system interconnection
OUI	organization unique identifier
PAE	port access entity
PDU	protocol data unit
PSK	preshared key
SME	station management entity
STAKeyID	STAKey index
STAKeySA	STAKey security association
UCK	unicast integrity check key
UEK	unicast encryption key
USK	unicast session key
USKID	unicast session key index
USKSA	unicast session key security association
WAI	WLAN authentication infrastructure
WAPI	WLAN authentication and privacy infrastructure
WEP	wired equivalent privacy

WM	wireless medium
WPI	WLAN privacy infrastructure

5. General description

5.1 General description of the architecture

This subclause presents the concepts and terminology used within the ISO/IEC 8802-11:2006 document. Specific terms are defined in Clause 3. Illustrations convey key ISO/IEC 8802-11 concepts and the interrelationships of the architectural components. ISO/IEC 8802-11 uses an architecture to describe functional components of an ISO/IEC 8802-11 LAN. The architectural descriptions are not intended to represent any specific physical implementation of ISO/IEC 8802-11.

5.1.1 How wireless LAN systems are different

Wireless networks have fundamental characteristics that make them significantly different from traditional wired LANs. Some countries impose specific requirements for radio equipment in addition to those specified in this standard.

5.1.1.1 Destination address does not equal destination

See 5.1.1.1 of ISO/IEC 8802-11:2006.

5.1.1.2 The media impact the design

See 5.1.1.2 of ISO/IEC 8802-11:2006.

5.1.1.3 The impact of handling mobile stations

See 5.1.1.3 of ISO/IEC 8802-11:2006.

5.1.1.4 Interaction with other IEEE 802 layers

ISO/IEC 8802-11 is required to appear to higher layers [logical link control (LLC)] as a current style IEEE 802 LAN. This requires that the ISO/IEC 8802-11 network handle station mobility within the MAC sublayer. To meet reliability assumptions (that LLC makes about lower layers), it is necessary for ISO/IEC 8802-11 to incorporate functionality that is untraditional for MAC sublayers.

In a WLAN authentication and privacy infrastructure (WAPI) security network, WLAN authentication infrastructure (WAI) is used to implement mutual authentication and provides a WAI Controlled Port. WLAN privacy infrastructure (WPI) provides functions to protect data frames. All STAs have a corresponding WAI Port Access Entity (PAE) that handles these services.

5.1.1.5 Interaction with non-IEEE 802 protocols

In a WAPI security network, WAI protocols, which are non-IEEE 802 protocols, are utilized for authentication and key management (AKM) services.

5.2 Components of the ISO/IEC 8802-11 architecture

The ISO/IEC 8802-11 architecture consists of several components that interact to provide a wireless LAN that supports station mobility transparently to upper layers.

The basic service set (BSS) is the basic building block of an ISO/IEC 8802-11 LAN. Figure 1 shows two BSSs, each of which has two stations that are members of the BSS.

It is useful to think of the ovals used to depict a BSS as the coverage area within which the member stations of the BSS may remain in communication. (The concept of area, while not precise, is often good enough.) If a station moves out of its BSS, it can no longer directly communicate with other members of the BSS.

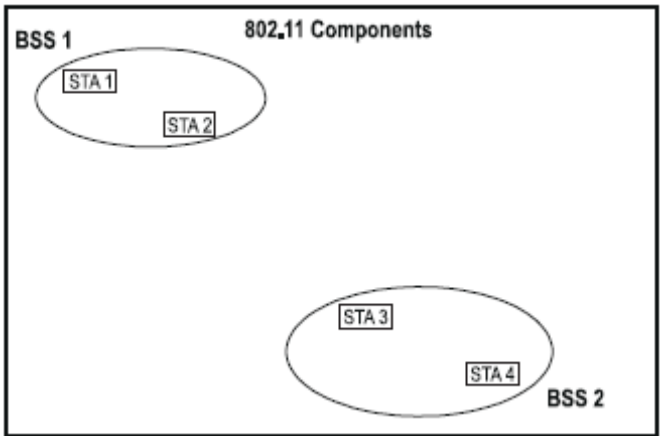

Figure 1—Basic service sets

5.2.1 The independent BSS as an ad hoc network

See 5.2.1 of ISO/IEC 8802-11:2006.

5.2.2 Distribution system concepts

PHY limitations determine the direct station-to-station distance that may be supported. For some networks this distance is sufficient; for other networks, increased coverage is required.

Instead of existing independently, a BSS may also form a component of an extended form of network that is built with multiple BSSs. The architectural component used to interconnect BSSs is the *distribution system* (DS).

ISO/IEC 8802-11 logically separates the wireless medium (WM) from the distribution system medium (DSM). Each logical medium is used for different purposes, by a different component of the architecture. The ISO/IEC

8802-11 definitions neither preclude, nor demand, that the multiple media be either the same or different.

Recognizing that the multiple media are *logically* different is key to understanding the flexibility of the architecture. The ISO/IEC 8802-11 LAN architecture is specified independently of the physical characteristics of any specific implementation.

The DS enables mobile device support by providing the logical services necessary to handle address to destination mapping and seamless integration of multiple BSSs.

An access point (AP) is a STA that provides access to the DS by providing DS services in addition to acting as a STA.

Figure 2 adds the DS and AP components to the ISO/IEC 8802-11 architecture picture.

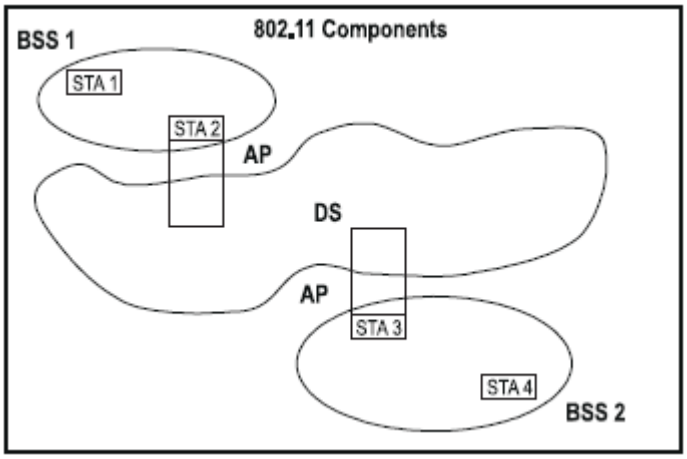

Figure 2—Distribution systems and access points

Data move between a BSS and the DS via an AP. Note that all APs are also STAs; thus they are addressable entities. The addresses used by an AP for communication on the WM and on the DSM are not necessarily the same.

5.2.2.1 Extended service set (ESS): The large coverage network

See 5.2.2.1 of ISO/IEC 8802-11:2006.

5.2.2.2 WAPI

WAPI defines a number of security features different from the RSNA. These features include the following:

— mutual authentication mechanisms for STAs
— Key management algorithms
— Cryptographic key establishment
— An enhanced data encapsulation mechanism, called WPI

WAPI relies on two components external to the ISO/IEC 8802-11 architecture.

The first component is a WAI port access entity (PAE). PAEs are present on all STAs and control the forwarding of data to and from the medium access control (MAC). An access point (AP) always implements an Authenticator PAE and a STA always implements a Supplicant PAE. In an independent basic service set (IBSS), each STA implements both an Authenticator PAE and a Supplicant PAE.

The second component is the Authentication Server (AS). The AS may authenticate the AP and the STA. More importantly, the AS can make the AP and the STA authenticate each other by performing a tri-element peer authentication (TePA) protocol. In certain applications, the AS may be integrated into the same physical device as the AP.

5.3 Logical service interfaces

See 5.3 of ISO/IEC 8802-11:2006.

5.4 Overview of the services

There are nine services specified by ISO/IEC 8802-11. Six of the services are used to support MSDU delivery between STAs. Three of the services are used to control ISO/IEC 8802-11 LAN access and confidentiality.

This subclause presents the services, an overview of how each service is used, and a description of how each service relates to other services and the ISO/IEC 8802-11 architecture. The services are presented in an order designed to help build an understanding of the operation of an ISO/IEC 8802-11 ESS network. As a result, the SSs and DSSs are intermixed in order (rather than being grouped by category).

Each of the services is supported by one or more MAC frame types. Some of the services are supported by MAC management messages and some by MAC data messages. All of the messages gain access to the WM via the ISO/IEC 8802-11 MAC sublayer medium access method specified in Clause 9.

The ISO/IEC 8802-11 MAC sublayer uses three types of messages—data, management, and control (see Clause 7). The data messages are handled via the MAC data service path.

MAC management messages are used to support the ISO/IEC 8802-11 services and are handled via the MAC management service data path.

MAC control messages are used to support the delivery of ISO/IEC 8802-11 data and management messages.

The examples here assume an ESS network environment. The differences between the ESS and the IBSS network environments are discussed separately in 5.6.

5.4.1 Distribution of messages within a DS

See 5.4.1 of ISO/IEC 8802-11:2006.

5.4.2 Services that support the distribution service

The primary purpose of a MAC sublayer is to transfer MSDUs between MAC sublayer entities. The information required for the distribution service to operate is provided by the association services. Before a data message can be handled by the distribution service, a STA shall be "associated."

To understand the concept of association, it is necessary first to understand the concept of mobility.

5.4.2.1 Mobility types

See 5.4.2.1 of ISO/IEC 8802-11:2006

5.4.2.2 Association

To deliver a message within a DS, the distribution service needs to know which AP to access for the given ISO/IEC 8802-11 STA. This information is provided to the DS by the concept of association. Association is necessary, but not sufficient, to support BSS-transition mobility. Association is sufficient to support no-transition mobility. Association is a DSS.

In a WAPI security network, this is different. In WAPI, the WAI Port determines when to allow data traffic across an ISO/IEC 8802-11 link. A single WAI Port maps to one association, and each association maps to an WAI Port. An WAI Port consists of an WAI Controlled Port and an WAI Uncontrolled Port. The WAI Controlled Port is blocked from passing general data traffic between two STAs until an WAI authentication procedure completes successfully over the WAI Uncontrolled Port. Once the AKM completes successfully, data protection is enabled to prevent unauthorized access, and the WAI Controlled Port unblocks to allow protected data traffic. WAI Supplicants and Authenticators exchange protocol information via the WAI Uncontrolled Port. It is expected that most other protocol exchanges will make use of the WAI Controlled Ports. However, a given protocol may need to bypass the authorization function and make use of the WAI Uncontrolled Port.

Before a STA is allowed to send a data message via an AP, it shall first become associated with the AP. The act of becoming associated invokes the association service, which provides the STA to AP mapping to the DS. The DS uses this information to accomplish its message distribution service. How the information provided by the association service is stored and managed within the DS is not specified by this standard.

At any given instant, a STA may be associated with no more than one AP. This ensures that the DS may determine a unique answer to the question, "Which AP is serving STA X?" Once an association is completed, a STA may make full use of a DS (via the AP) to communicate. Association is always initiated by the mobile STA, not the AP.

An AP may be associated with many STAs at one time.

A STA learns what APs are present and then requests to establish an association by invoking the association service. For details of how a station learns about what APs are present, see 11.1.3.

5.4.2.3 Reassociation

Association is sufficient for no-transition message delivery between ISO/IEC 8802-11 stations. Additional functionality is needed to support BSS-transition mobility. The additional required functionality is provided by the reassociation service. Reassociation is a DSS.

The reassociation service is invoked to "move" a current association from one AP to another. This keeps the DS informed of the current mapping between AP and STA as the station moves from BSS to BSS within an ESS. Reassociation also enables changing association attributes of an established association while the STA remains associated with the same AP. Reassociation is always initiated by the mobile STA.

No facilities are provided to move a WAPI process during reassociation, then the old WAPI process will be deleted, and a new WAPI process will need to be constructed.

5.4.2.4 Disassociation

See 5.4.2.4 of ISO/IEC 8802-11:2006.

5.4.3 Access and confidentiality control services

Two services are required for ISO/IEC 8802-11 to provide functionality equivalent to that which is inherent to wired LANs. The design of wired LANs assumes the physical attributes of wire. In particular, wired LAN design assumes the physically closed and controlled nature of wired media. The physically open medium nature of an ISO/IEC 8802-11 LAN violates those assumptions.

Two services are provided to bring the ISO/IEC 8802-11 functionality in line with wired LAN assumptions, authentication and confidentiality Authentication is used instead of the wired media physical connection. Confidentiality is used to provide the confidential aspects of closed wired media.

In a wireless LAN that does not support WAPI, two services, authentication and confidentiality, are defined. ISO/IEC 8802-11 authentication is used instead of the wired media physical connection. WEP encryption was defined to provide the confidentiality aspects of closed wired media.

In a WAPI security network, the WAI Controlled Port along with WAI protocols is used to provide access control. The ISO/IEC 8802-11 station management entity (SME) provides key management via an exchange of WAI protocol data frames. Confidentiality and data integrity are provided by WPI.

5.4.3.1 Authentication

ISO/IEC 8802-11 authentication operates at the link level between ISO/IEC 8802-11 STAs. ISO/IEC 8802-11 does not provide either end-to-end (message origin to message destination) or user-to-user authentication.

ISO/IEC 8802-11 attempts to control LAN access via the authentication service. ISO/IEC 8802-11 authentication is an SS. This service may be used by all STAs to establish their identity to STAs with which they

communicate, in both extended service set (ESS) and IBSS networks. If a mutually acceptable level of authentication has not been established between two STAs, an association shall not be established.

ISO/IEC 8802-11 defines two authentication methods: Open System authentication and Shared Key authentication.
Open System authentication admits any STA to the DS. Shared Key authentication relies on WEP to demonstrate knowledge of a WEP encryption key. The ISO/IEC 8802-11 authentication mechanism also allows definition of new authentication methods.

In a WAPI security network, WAI is used to implement mutual authentication between two STAs; it is established after authentication and association. In a BSS, when the authentication is successful, the STA can securely access the AP; otherwise, both the AP and STA refuse to connect with each other. In an IBSS, when the authentication is successful, the STA can securely access the peer STA; otherwise, both STAs refuse to communicate with each other. Detailed definition is in Clause 8.

In a BSS or an IBSS, WAI supports authentication based on certificates, or preshared keys. Certificate authentication utilizes the public key certificates to authenticate STAs and the AP.

In WAPI, the WAI ASUE and AE exchange protocol information via the WAI Uncontrolled Port. The WAI Controlled Port is blocked from passing general data traffic between two STAs until a WAI authentication procedure completes successfully over the WAI Uncontrolled Port.

In a WAPI ESS, Open System authentication is required. In a WAPI IBSS, Open System authentication is optional.

Management information base (MIB) functions are provided in Annex D to support the standardized authentication schemes.

A STA may be authenticated with many other STAs at any given instant.

5.4.3.2 Deauthentication

The deauthentication service is invoked when an existing Open System or Shared Key authentication is to be terminated. Deauthentication is an SS.

In an ESS, because authentication is a prerequisite for association, the act of deauthentication shall cause the station to be disassociated. The deauthentication service may be invoked by either authenticated party (non-AP STA or AP). Deauthentication is not a request; it is a notification. Deauthentication shall not be refused by either party. When an AP sends a deauthentication notice to an associated STA, the association shall also be terminated.

In a WAPI ESS, Open System authentication is required. In a WAPI ESS, deauthentication results in termination of any association for the deauthenticated station. It also results in the WAI Controlled Port for that STA being disabled and deletes the unicast session key security association (USKSA). The deauthentication notification is

provided to WAI via the MAC layer.

In WAPI, deauthentication also destroys any related USKSA, multicast session key security association (MSKSA), and STAKey security associations (STAKeySAs) that exist in the STA and closes the associated WAI Controlled Port. If base key (BK) caching is not enabled, deauthentication also destroys the base key security association (BKSA) from which the deleted USKSA was derived.

In a WAPI IBSS, Open System authentication is optional, but a STA is required to recognize Deauthentication frames. Deauthentication results in the WAI Controlled Port for that STA being disabled and deletes the USKSA.

5.4.3.3 Confidentiality

In a wired LAN, only those stations physically connected to the wire can send or receive LAN traffic. With a wireless shared medium, this is not the case. Any ISO/IEC 8802-11-compliant STA can receive all like-PHY ISO/IEC 8802-11 traffic that is within range and can transmit to any other ISO/IEC 8802-11 STA within range. Thus the connection of a single wireless link (without confidentiality) to an existing wired LAN may seriously degrade the security level of the wired LAN.

To bring the security of the wireless LAN up to the level implicit in wired LAN design, ISO/IEC 8802-11 provides the ability to protect the contents of messages. This functionality is provided by the confidentiality service. Confidentiality is an SS.

ISO/IEC 8802-11 provides three cryptographic algorithms to protect data traffic: WEP, TKIP, and CCMP. WEP and TKIP are based on the RC48 algorithm, and CCMP is based on the advanced encryption standard (AES). A means is provided for STAs to select the algorithm(s) to be used for a given association.

This standard specifies an optional confidentiality algorithm, WPI, that is designed to satisfy the goal of wired LAN “equivalent” confidentiality. The algorithm is not designed for ultimate security but rather to be “at least as secure as a wire” See Clause 8 for more details.

This standard uses the WPI mechanism (see Clause 8) to perform the actual encryption of messages. MIB functions are provided to support WPI.

Note that confidentiality may only be invoked for data frames. All stations initially start “in the clear” in order to set up the authentication, identity authentication and confidentiality services.

The default confidentiality state for all ISO/IEC 8802-11 STAs is “in the clear.” If the confidentiality service is not invoked, all messages shall be sent unprotected. If this policy is unacceptable to the sender, it shall not send data frames; and if the policy is unacceptable to the receiver, it shall discard any received data frames, as well as protected data frames using a key not available at the receiving station, are discarded without an indication to LLC (or without indication to distribution services in the case of “To DS” frames received at an AP). These frames are acknowledged on the WM [if received without frame check sequence (FCS) error] to avoid wasting WM bandwidth on retries.

5.5 Relationships between services

See 5.5 of ISO/IEC 8802-11:2006.

5.6 Differences between ESS and IBSS LANs

See 5.6 of ISO/IEC 8802-11:2006.

5.7 Message information contents that support the services

Each service is supported by one or more ISO/IEC 8802-11 messages. Information items are given by name; for corresponding values, see Clause 7.

5.7.1 Data

See 5.7.1 of ISO/IEC 8802-11:2006.

5.7.2 Association

See 5.7.2 of ISO/IEC 8802-11:2006.

5.7.3 Reassociation

See 5.7.3 of ISO/IEC 8802-11:2006.

5.7.4 Disassociation

See 5.7.4 of ISO/IEC 8802-11:2006.

5.7.5 Confidentiality

See 5.7.5 of ISO/IEC 8802-11:2006.

5.7.6 Authentication

In a WAPI security network, for a STA to authenticate with another STA using Open System, Shared Key or Public-key Certificate authentication. The authentication service causes one or more authentication management frames to be exchanged. The exact sequence of frames and their content are dependent on the authentication scheme invoked. For all of these authentication schemes, the authentication algorithm is identified within the management frame body.

In an IBSS environment, either station may be the initiating STA (STA 1). In an ESS environment, STA 1 is the mobile STA, and STA 2 is the AP.

This standard defines one subtype of authentication service: Open System. The subtype invoked is indicated in the body of authentication management frames. Thus authentication frames are selfidentifying with respect to authentication algorithm. All management frames of subtype authentication shall be unicast frames as authentication is performed between pairs of stations (i.e., multicast authentication is not allowed). Management frames of subtype Deauthentication are advisory, and may therefore be sent as group-addressed frames. A mutual authentication relationship shall exist between two stations following a successful authentication exchange as described below. Authentication shall be used between stations and the AP in an infrastructure BSS. Authentication may be used between two STAs in an IBSS.

Open System authentication is the simplest of the available authentication algorithms. Essentially it is a null authentication algorithm. Any STA that requires authentication with this algorithm may become authenticated if dot11Authentication Type at the recipient station is set to Open System authentication. Open System authentication is not required to be successful as a STA may decline to authentication with any particular other STA. Open System authentication is the default authentication algorithm. Open System authentication involves a two-step authentication transaction sequence. The first step in the sequence is the identity assertion and request for authentication. The second step in the sequence is the authentication result. If the result is successful, the STAs shall be mutually authenticated.

Open System authentication (first frame)

—Message type: Management

—Message subtype: authentication

— Information items:

- Authentication Algorithm Identification = .Open System.
- Station Identity Assertion (in SA field of header)
- Authentication transaction sequence number = 1
- Authentication algorithm dependent information (none)

—Direction of message: From authentication initiating STA to authenticating STA

Open System authentication (final frame)

— Message type: Management

—Message subtype: Authentication

—Information items:

- Authentication Algorithm Identification = .Open System.
- Authentication transaction sequence number = 2
- Authentication algorithm dependent information (none)
- The result of the requested authentication as defined in 7.3.1.9

— Direction of message: From authenticating STA to initiating STA

5.7.7 Deauthentication

See 5.7.7 of ISO/IEC 8802-11:2006.

5.8 Reference model

This standard presents the architectural view, emphasizing the separation of the system into two major parts: the MAC of the data link layer and the PHY. These layers are intended to correspond closely to the lowest layers of the ISO/IEC basic reference model of Open Systems Interconnection (OSI) (ISO/IEC 7498-1: 1994[5]). The layers and sublayers described in this standard are shown in Figure 11.

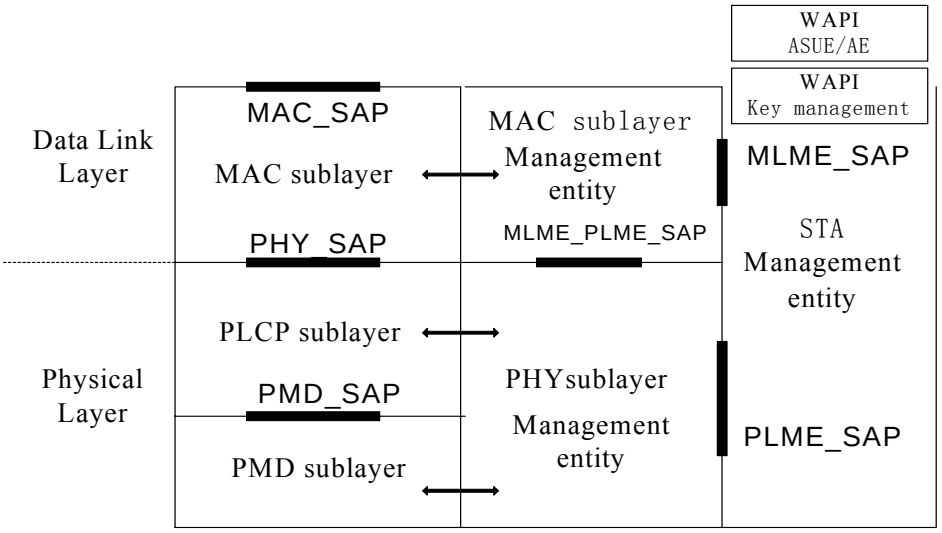

Figure 11—Portion of the ISO/IEC basic reference model covered in this standard

5.9 Establishing the security association

5.9.1 Infrastructure mode

The STA obtains the security policy of the AP by passively monitoring the Beacon frame or probing the AP on its own initiative (Figure 11a). If the WAI certificate Authentication and Key Management Protocol (AKMP) is applied, the AP sends an authentication activation packet to start the certificate authentication process. After this process, the unicast key negotiation process and the multicast key announcement process are executed between the AP and STA. If the PSK mechanism is applied, the unicast key negotiation process and the multicast key announcement process are directly executed between the AP and STA.

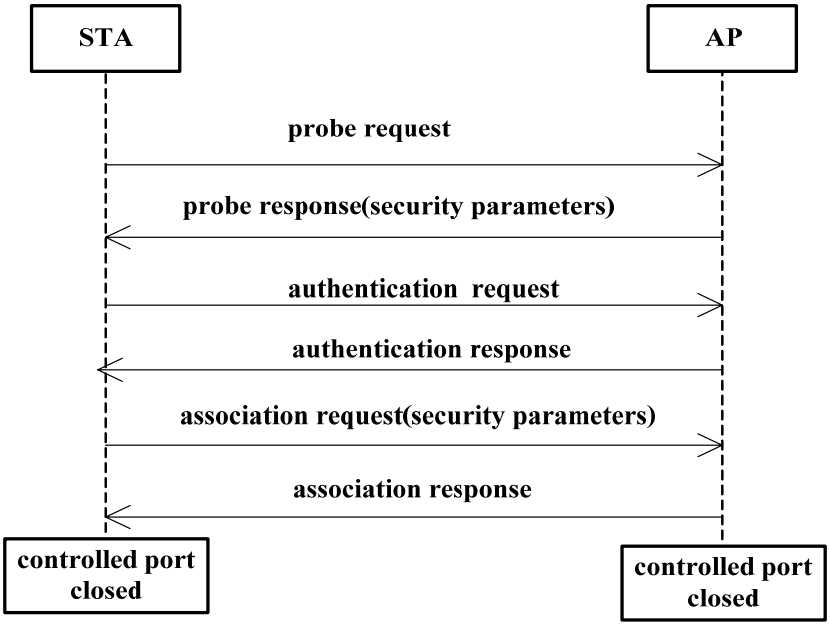

Figure 11a—The establishment of security association in infrastructure mode

5.9.2 IBSS mode

The initiation of the PSK key negotiation process in an IBSS :

In an IBSS, there are two five-step handshakes (the first three steps are the unicast key negotiation process and the last two are the multicast key announcement process) between two STAs. Two independent unicast keys will be negotiated. And their own multicast key will be announced to each other. In the course of practical communication, for a pair of STAs, the unicast data between the two STAs are encrypted or decrypted with the unicast encryption key that is generated by the process initiated by the STA (as AE) with the larger MAC address, whereas the keys generated by the process initiated by the STA (as AE) with the smaller MAC address will not be used. Every STA encrypts its broadcast/multicast data with the multicast key announced by itself before sending them and the other STAs decrypt these data with the same key when receiving them.

The reasons or conditions for the initiation of the unicast key negotiation process based on the PSK are listed as below:

a) When a STA receives a Beacon frame or a Probe Response frame from another STA which has not completed a unicast key negotiation process with it.
b) When a STA receives the first protocol packet of the unicast key negotiation process from another STA which has not completed a process of unicast key negotiation with it.
c) When a STA receives a WPI encapsulation broadcast/multicast data, if it has not the corresponding decryption key, it shall notify the station management entity with an MLME-PROTECTEDFRAMEDROPPED.indication primitive. If the entity receives this primitive and has not completed a unicast key negotiation with the peer STA specified in the primitive.

The initiation of authentication based on the certificate in an IBSS:

When the authentication method based on a certificate is chosen between two STAs, each STA initiates its own certificate authentication process, unicast key negotiation process and multicast key announcement process, respectively. After these processes, they establish two independent BKs as well as two independent USKs and announce their own multicast keys to each other. In the course of data communication, for a pair of STAs, the unicast data between the two STAs are encrypted or decrypted with the unicast encryption key that are generated from the process of unicast key negotiation initiated by the STA (as AE) with the larger MAC address, whereas the keys generated by the process initiated by the STA (as AE) with the smaller MAC address will not be used. Every STA encrypts its broadcast/multicast data with the multicast key announced by itself when sending them and the other STAs decrypt these data with the same key when receiving them.

The reasons or conditions of the initiation of the authentication process based on certificate are listed as below:

a) When a STA receives a Beacon frame or a Probe Response frame from another STA which has not completed a certificate authentication process with it.
b) When a STA receives the first protocol packet of the certificate authentication process from another STA which has not completed a process of certificate authentication with it.
c) When a STA receives a WPI encapsulation broadcast/multicast data, if it has not the corresponding decryption key, it shall notify the station management entity with an MLME-PROTECTEDFRAMEDROPPED.indication primitive. If the entity receives this primitive and has not completed a certificate authentication with the STA described in the primitive.

If the WAPI security parameters of the peer STA are unknown to another STA, the STA should send a Probe Request frame to the peer STA before the initiation of unicast key negotiation process. Then the security mechanism can be obtained from the Probe Response frame of the peer STA.

6. MAC service definition

6.1 Overview of MAC services

6.1.1 Asynchronous data service

See 6.1.1 of ISO/IEC 8802-11:2006.

6.1.2 Security services

Security services in this standard are provided by the authentication services and the WPI mechanism. The scope of the security services provided is limited to station-to-station data exchange. The confidentiality service offered is the encryption of the MSDU. For the purposes of this standard, WPI is viewed as a logical service located within the MAC sublayer as shown in the reference model, Figure 11. Actual implementations of the WPI service are transparent to the LLC and other layers above the MAC sublayer.

The security services provided by this part are as follows:

a) Confidentiality;
b) Authentication; and
c) Access control in conjunction with layer management.

During the authentication exchange, parties A and B exchange authentication information as described in Clause 8.

The MAC sublayer security services provided by WPI rely on information from nonlayer-2 management or system entities. Management entities communicate information to WPI through a set of MIB attributes.

6.1.3 MSDU ordering

See 6.1.3 of ISO/IEC 8802-11:2006.

6.1.4 MAC Data Service Infrastructure

MAC data plane architecture（i.e., processed that involve transport of all or part of an MSDU process）is shown in Figure 11b.

During transmission, an MSDU goes through some or all of the following processes: frame delivery deferral during power save mode, sequence number assignment, fragmentation, encryption, integrity protection, and frame formation. WAPI may block the MSDU at the Controlled Port.

During reception, a received data frame goes through processed of MPDU header + FCS validation, duplicate removal, decryption, defragmentation, integrity checking, and replay detection. The WAPI Controlled/ Uncontrolled Ports discard the MSDU if the Controlled Port is not enabled or if the MSDU does not represent a WAPI frame. WPI MPDU frame order enforcement occurs prior to MSDU defragmentation; therefore, defragmentation will fail if MPDUs arrive out of order.

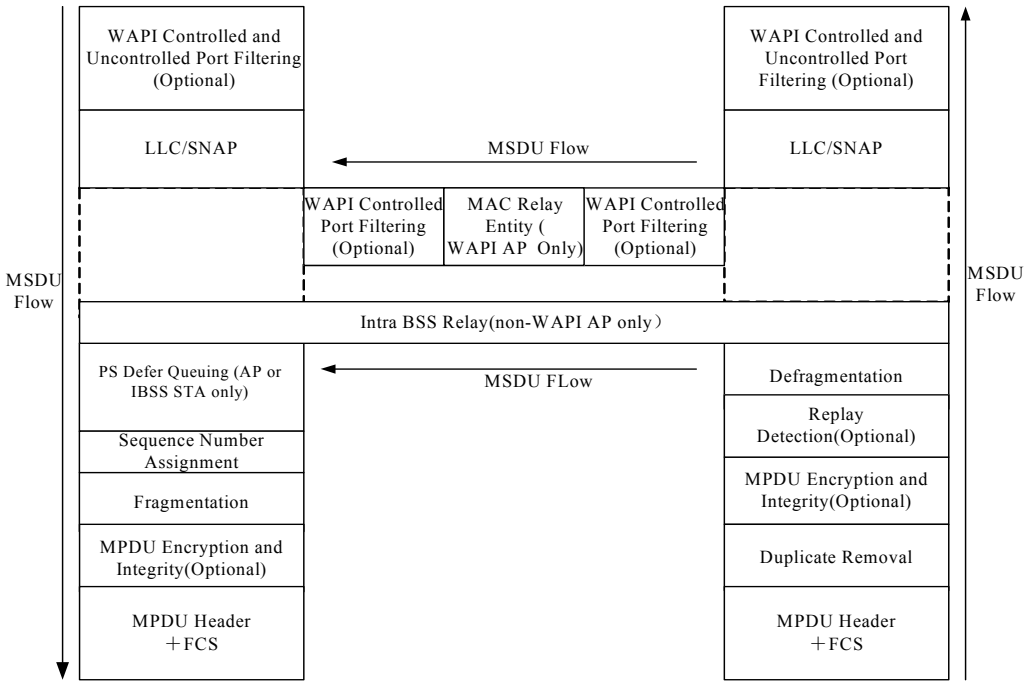

Figure 11b MAC data plane architecture

6.2 Detailed service specification

See 6.2 of ISO/IEC 8802-11:2006.

7. Frame formats

The format of the MAC frames is specified in this clause. All stations shall be able to properly construct frames for transmission and decode frames upon reception, as specified in this clause.

7.1 MAC frame formats

See 7.1.1 of ISO/IEC 8802-11:2006.

7.2 Format of individual frame types

7.2.1 Control frames

See 7.2.1 of ISO/IEC 8802-11:2006.

7.2.2 Data frames

See 7.2.2 of ISO/IEC 8802-11:2006.

7.2.3 Management frames

The frame format for a Management frame is independent of frame subtype and is as defined in Figure 23.

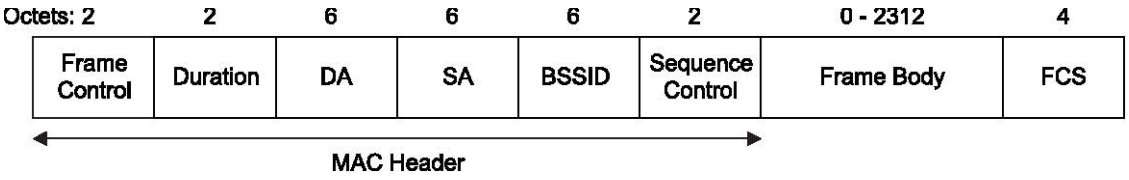

Figure 23—Management frame format

A STA uses the contents of the Address 1 field to perform the address matching for receive decisions. In the case where the Address 1 field contains a group address and the frame type is other than Beacon, the BSSID also is validated to ensure that the broadcast or multicast originated in the same BSS. If the frame type is Beacon, other address matching rules apply, as specified in 11.1.2.3.

The address fields for management frames do not vary by frame subtype.

The BSSID of the management frame is determined as follows:

a) If the station is an AP or is associated with an AP, the BSSID is the address currently in use by the STA contained in the AP.

b) If the station is a member of an IBSS, the BSSID is the BSSID of the IBSS.

c) In Management frames of subtype Probe Request, the BSSID is either a specific BSSID, or the broadcast BSSID as defined in the procedures specified in Clause 10.

The DA is the destination of the frame.

The SA is the address of the station transmitting the frame.

Within all management type frames sent during the CFP, the Duration field is set to the value 32 768. Within all management type frames sent during the contention period, the Duration field is set according to the following rules:

— If the DA field contains a group address, the duration value is set to 0.

— If the More Fragments bit is set to 0 in the Frame Control field of a frame and the DA contains an individual address, the duration value is set to the time, in microseconds, required to transmit one ACK frame, plus one SIFS interval.

— If the More Fragments bit is set to 1 in the Frame Control field of a frame, and the DA contains an individual address, the duration value is the time, in microseconds, required to transmit the next fragment of this management frame, plus two ACK frames, plus three SIFS intervals.

The duration value calculation for the management frame is based on the rules in 9.6 that determine the data rate at which the control frames in the frame exchange sequence are transmitted. If the calculated duration includes a fractional microsecond, that value is rounded up to the next higher integer. All stations process Duration field values less than or equal to 32 767 from valid management frames to update their NAV settings as appropriate under the coordination function rules.

The frame body consists of the fixed fields and information elements defined for each management frame subtype. All fixed fields and information elements are mandatory unless stated otherwise, and they can appear only in the specified order. Stations encountering an element type they do not understand ignore that element. Element type codes not explicitly defined in this standard are reserved, and do not appear in any frames.

7.2.3.1 Beacon frame format

The frame body of a management frame of subtype Beacon contains the information shown in Table 5.

Table 5—Beacon frame body

Order	Information	Notes
1	Timestamp	
2	Beacon interval	
3	Capability information	
4	SSID	

5	Supported rates	
6	FH Parameter Set	The FH Parameter Set information element is present within Beacon frames generated by STAs using frequency-hopping PHYs.
7	DS Parameter Set	The DS Parameter Set information element is present within Beacon frames generated by STAs using direct sequence PHYs.
8	CF Parameter Set	The CF Parameter Set information element is only present within Beacon frames generated by APs supporting a PCF.
9	IBSS Parameter Set	The IBSS Parameter Set information element is only present within Beacon frames generated by STAs in an IBSS.
10	TIM	The TIM information element is only present within Beacon frames generated by APs.
21	RSN	The RSN information element is only present within Beacon frames generated by STAs that have dot11RSNAEnabled set to TRUE.
22	WAPI Parameter Set	The WAPI Parameter Set information element is only present within Beacon frames generated by STAs when the WAPI mechanism is enabled.

7.2.3.2 IBSS Announcement Traffic Indication Message (ATIM) frame format

See 7.2.3.2 of ISO/IEC 8802-11:2006.

7.2.3.3 Disassociation frame format

See 7.2.3.3 of ISO/IEC 8802-11:2006.

7.2.3.4 Association Request frame format

The frame body of a management frame of subtype Association Request contains the information shown in Table 7.

Table 7 —Association Request frame body

1	Capability information	
2	Listen interval	
3	SSID	
4	Supported rates	
8	RSN	The RSN information element is only present within Association Request frames generated by STAs that have dot11RSNAEnabled set to TRUE.
9	WAPI Parameter Set	The WAPI Parameter Set information element is only present within Association request frames generated by STAs when the WAPI mechanism is enabled.

7.2.3.5 Association Response frame format

See 7.2.3.5 of ISO/IEC 8802-11:2006.

7.2.3.6 Reassociation Request frame format

The frame body of a management frame of subtype Reassociation Request contains the information shown in Table 9.

Table 9—Reassociation Request frame body

1	Capability information	
2	Listen interval	
3	Current AP address	
4	SSID	
5	Supported rates	
9	RSN	The RSN information element is only present within Reassociation Request frames generated by STAs that have dot11RSNAEnabled set to TRUE.
10	WAPI Parameter Set	The WAPI Parameter Set information element is only present within Reassociation request frames generated by STAs when the WAPI mechanism is enabled.

7.2.3.7 Reassociation Response frame format

See 7.2.3.7 of ISO/IEC 8802-11:2006.

7.2.3.8 Probe Request frame format

See 7.2.3.8 of ISO/IEC 8802-11:2006.

7.2.3.9 Probe Response frame format

The frame body of a management frame of subtype Probe Response contains the information shown in Table 12.

Table 12—Probe Response frame body

Order	Information	Notes
1	Timestamp	
2	Beacon interval	
3	Capability information	
4	SSID	
5	Supported rates	
6	FH Parameter Set	The FH Parameter Set information element is present within Probe Response frames generated by STAs using frequency-hopping PHYs.
7	DS Parameter Set	The DS Parameter Set information element is present within Probe Response frames generated by STAs using direct sequence PHYs.
8	CF Parameter Set	The CF Parameter Set information element is only present within Probe Response frames generated by APs supporting a PCF.
9	IBSS Parameter Set	The IBSS Parameter Set information element is only present within Probe Response frames generated by STAs in an IBSS.
21	RSN	The RSN information element is only present within Probe Response frames generated by STAs that have dot11RSNAEnabled set to TRUE.
22	WAPI Parameter Set	The WAPI Parameter Set information element is only present within Probe Response frames generated by STAs when the WAPI mechanism is enabled.
23-n	Requested information elements	Elements requested by the Request information element of the Probe Request frame.

7.2.3.10 Authentication frame format

See 7.2.3.10 of ISO/IEC 8802-11:2006

7.2.3.11 Deauthentication

See 7.2.3.11 of ISO/IEC 8802-11:2006.

7.3 Management frame body components

Within management frames, fixed-length mandatory frame body components are defined as fixed fields; variable length mandatory and all optional frame body components are defined as information elements.

7.3.1 Fixed fields

7.3.1.1 Authentication Algorithm Number field

See 7.3.1.1 of ISO/IEC 8802-11:2006.

7.3.1.2 Authentication Transaction Sequence Number field

See 7.3.1.2 of ISO/IEC 8802-11:2006.

7.3.1.3 Beacon Interval field

See 7.3.1.3 of ISO/IEC 8802-11:2006.

7.3.1.4 Capability Information field

The Capability Information field contains a number of subfields that are used to indicate requested or advertised capabilities. The length of the Capability Information field is 2 octets. The Capability Information field consists of following subfields: ESS, IBSS, CF-Pollable, CF-Poll Request and Privacy. The remaining part of Capability Information field is reserved. The format of the Capability Information field is as illustrated in Figure 27.

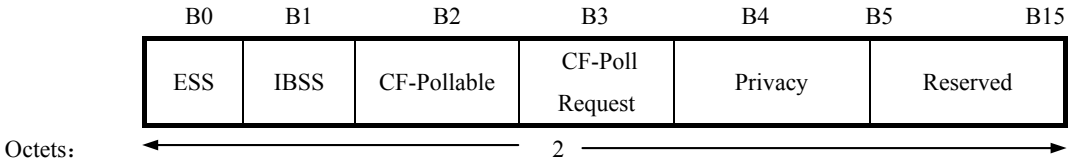

Figure 27 Capability Information fixed field

Each Capability Information subfield is interpreted only in the management frame subtypes for which the transmission rules are defined.

APs set the ESS subfield to 1 and the IBSS subfield to 0 within transmitted Beacon or Probe Response management frames. STAs within an IBSS set the ESS subfield to 0 and the IBSS subfield to 1 in transmitted Beacon or Probe Response management frames.

STAs set the CF-Pollable and CF-Poll Request subfields in Association and Reassociation Request management frames according to Table 16.

Table 16 STA usage of CF-Pollable and CF-Poll Request

CF-Pollable	CF-Poll request	Meaning
0	0	STA is not CF-Pollable
0	1	STA is CF-Pollable, not requesting to be placed on the CF-Polling list
1	0	STA is CF-Pollable, requesting to be placed on the CF-Polling list
1	1	STA is CF-Pollable, requesting never to be polled

APs set the CF-Pollable and CF-Poll Request subfields in Beacon, Probe Response, Association Response, and Reassociation Response management frames according to Table 17. An AP sets the CF-Pollable and CF-Poll Request subfield values in Association Response and Reassociation Response management frames equal to the values in the last Beacon or Probe Response frame that in transmitted.

Table 17 AP usage of CF-Pollable and CF-Poll Request

CF-Pollable	CF-Poll request	Meaning
0	0	No point coordinator at AP
0	1	Point coordinator at AP for delivery only (no polling)
1	0	Point coordinator at AP for delivery and polling
1	1	Reserved

APs set the Privacy subfield to 1 within transmitted Beacon, Probe Response, Association Response, and Reassociation Response management frames if WAPI encryption is required for all data type frames exchanged within the BSS. If WAPI encryption is not required, the Privacy subfield is set to 0.

STAs within a BSS which are not APs set the Privacy subfield to 0 in transmitted Association or Reassociation management frames. APs neglect the Privacy subfield of Association or Reassociation management frames.

STAs within an IBSS set the Privacy subfield to 1 in transmitted Beacon or Probe Response management frames if WAPI encryption is required for all data type frames exchanged within the IBSS. If WAPI encryption is not required, the Privacy subfield is set to 0.

If the WAPI information elements are contained in transmitted Beacon or Probe Response management frames, the Privacy subfield is set to 0.

7.3.1.5 Current AP Address field

See 7.3.1.5 of ISO/IEC 8802-11:2006.

7.3.1.6 Listen Interval field

See 7.3.1.6 of ISO/IEC 8802-11:2006.

7.3.1.7 Reason Code field

This Reason Code field is used to indicate the reason that an unsolicited notification management frame of type Disassociation or Deauthentication was generated. The length of the Reason Code field is 2 octets. The Reason Code field is illustrated in Figure 30.

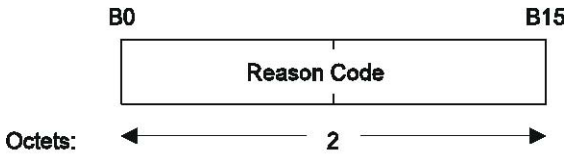

Figure 30—Reason Code fixed field

The reason codes are defined in Table 18.

Table 18—Reason codes

Reason code	Meaning
0	Reserved
1	Unspecified reason
2	Previous authentication no longer valid
3	Deauthentication because sending station is leaving (or has left) IBSS or ESS
4	Disassociated due to inactivity
5	Disassociated because AP is unable to handle all currently associated stations
6	Class 2 frame received from nonauthenticated station
7	Class 3 frame received from nonauthenticated station
8	Disassociated because sending station is leaving (or has left) BSS
9	Station requesting (re)association is not authenticated with responding station
10-12	Reserved
13	Invalid information element
14	MIC failure
15	4-Way Handshake timeout
16	Group Key Handshake timeout
17	Information element in 4-Way Handshake different from (Re) Association Request/Probe Response/Beacon frame
18	Invalid group cipher
19	Invalid pairwise cipher
20	Invalid AKMP
21	Unsupported RSN information element version

22	Invalid RSN information element capabilities
23	IEEE 802.1X authentication failed
24	Cipher suite rejected per security policy
25	Unicast Key Negotiation timeout
26	Multicast Key Announcement timeout
27	Information element in Unicast Key Negotiation different from (Re)Association Request/Probe Response/Beacon frame
28	Invalid Multicast cipher
29	Invalid Unicast cipher
30	Unsupported WAPI information element version
31	Invalid WAPI information element capabilities
32	WAI certificate authentication failed
33–65 535	Reserved

7.3.1.8 Association ID (AID) field

See 7.3.1.8 of ISO/IEC 8802-11:2006.

7.3.1.9 Status Code field

The Status Code field is used in a response management frame to indicate the success or failure of a requested operation. The length of the Status Code field is 2 octets. The Status Code field is illustrated in Figure 32.

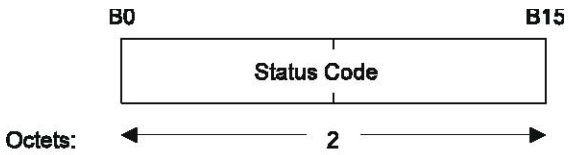

Figure 32—Status Code fixed field

If an operation is successful, then the status code is set to 0. If an operation results in failure, the status code indicates a failure cause. The failure cause codes are defined in Table 19.

Table 19—Status codes

Status code	Meaning
0	Successful
1	Unspecified failure
2–9	Reserved
10	Cannot support all requested capabilities in the Capability Information field
11	Reassociation denied due to inability to confirm that association exists
12	Association denied due to reason outside the scope of this standard
13	Responding station does not support the specified authentication algorithm
14	Received an Authentication frame with authentication transaction sequence number out of expected sequence
15	Authentication rejected because of challenge failure
16	Authentication rejected due to timeout waiting for next frame in sequence
17	Association denied because AP is unable to handle additional associated stations
18	Association denied due to requesting station not supporting all of the data rates in the BSSBasicRateSet parameter
19	Association denied due to requesting station not supporting the Short Preamble option
20	Association denied due to requesting station not supporting the PBCC Modulation option
21	Association denied due to requesting station not supporting the Channel Agility option
22-39	Reserved
40	Invalid information element

41	Invalid group cipher
42	Invalid pairwise cipher
43	Invalid AKMP
44	Unsupported RSN information element version
45	Invalid RSN information element capabilities
46	Cipher suite rejected per security policy
47	Invalid unicast cipher
48	Invalid multicast cipher
49	Unsupported WAPI information element version
50	Invalid WAPI information element capabilities
51–65 535	Reserved

7.3.1.10 Timestamp field

See 7.3.1.10 of ISO/IEC 8802-11:2006.

7.3.2 Information elements

Elements are defined to have a common general format consisting of a 1 octet Element ID field, a 1 octet length field, and a variable-length element-specific information field. Each element is assigned a unique Element ID as defined in this standard. The Length field specifies the number of octets in the Information field. See Figure 34.

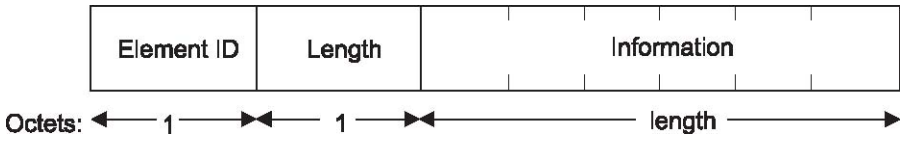

Figure 34—Element format

The set of valid elements is defined in Table 20.

Table 20—Element IDs

Information element	Element ID
SSID	0
Supported rates	1
FH Parameter Set	2

DS Parameter Set	3
CF Parameter Set	4
TIM	5
IBSS Parameter Set	6
Reserved	7–15
Challenge text	16
Reserved for challenge text extension	17–31
Reserved	32-47
RSN	48
Reserved	49-67
WAPI	68
Reserved	69–255

7.3.2.1 Service Set Identity (SSID) element

See 7.3.2.1 of ISO/IEC 8802-11:2006.

7.3.2.2 Supported Rates element

See 7.3.2.2 of ISO/IEC 8802-11:2006.

7.3.2.3 FH Parameter Set element

See 7.3.2.3 of ISO/IEC 8802-11:2006.

7.3.2.4 DS Parameter Set element

See 7.3.2.4 of ISO/IEC 8802-11:2006.

7.3.2.5 CF Parameter Set element

See 7.3.2.5 of ISO/IEC 8802-11:2006.

7.3.2.6 TIM

See 7.3.2.6 of ISO/IEC 8802-11:2006.

7.3.2.7 IBSS Parameter Set element

See 7.3.2.7 of ISO/IEC 8802-11:2006.

7.3.2.8 Challenge Text element

See 7.3.2.8 of ISO/IEC 8802-11:2006.

7.3.2.9 WAPI information element

The WAPI Parameter Set information element contains authentication and unicast cipher suite selectors, See Figure 41a. All STAs implementing WAPI shall support this element. The size of the WAPI information element is limited by the size of an information element, which is 255 octets.

	Element Identifier ID	Length	Version	Authentication and key Management (AKM) Suite Count	Authentication and key Management (AKM) Suite	Unicast Cipher Suite count	Unicast Cipher Suite	Multicast Cipher Suite	WAPI Capability Information	BKID Count	BKID List
Octets:	1	1	2	2	4×m	2	4×n	4	2	2	16×s

Figure 41a—WAPI Information Element format

In Figure 41a, m denotes AKM suite count, n is the unicast cipher suite count, and s is the BKID count.

— Element Identifier ID shall be 68 decimal.
— The Length field gives the number of octets of all the fields except the Element Identifier ID field and the Length field of the information element.
— The Version field gives the version number of the WAPI protocol. In this document, this field is set to 1, and other values are reserved.
— The Authentication and Key Management (AKM) Suite Count field denotes the number of the authentication and key management mechanisms that the STA supports.
— The Authentication and Key Management (AKM) Suite field contains the authentication and key management mechanisms that the STA supports. Here, m is the value of the Authentication and Key Management Suite Count field.
— The Unicast Cipher Suite Count field denotes the number of the unicast cipher algorithms that the STA supports.
— The Unicast Cipher Suite field contains the unicast cipher algorithms that STA supports. Here, n is the value of the Unicast Cipher Suite Count field.
— The Multicast Cipher Suite field contains the multicast cipher algorithms that the STA supports.
— WAPI Capability Information.

The WAPI Capability Information field indicates requested or advertised capabilities. Every subfield of the WAPI Capability Information field can be set to 0. The WAPI Capability Information field is 2 octets in length, and its subfields are defined as follows:

Figure 41b—Pre-authentication

Bit 0: Pre-Authentication. An AP sets the pre-authentication subfield of the WAPI Capabilities field to 1 when it

supports pre-authentication. Otherwise, it will be set to 0. A non-AP STA sets the pre-authentication subfield to 0.

— The BKID Count and list fields

The BKID Count and the BKID List fields are only present within Association or Reassociation Request frames sent to AP. The BKID Count field indicates the number of BKIDs in the BKID List field. The BKID List field contains 0 or more valid BKIDs that the STA believes to be valid for the destination AP. The BKIDs can refer to:

a) BKIDs generated by pre-authentication between the current STA and the destination AP
b) BKIDs generated by the certificate authentication
c) BKIDs generated by the preshared key

7.3.2.9.1 Authentication and Key Management Suite

A suite selector has the format shown in Figure 41c.

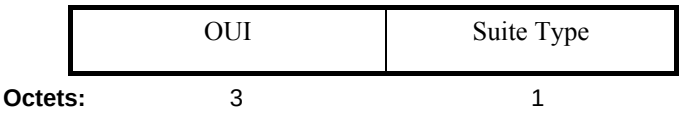

Figure 41c—Suite selector format

The suites defined in this specification are shown in Table 19a.

Table 19a—Authentication and Key Management Suites

OUI 3 octets	Suite Type 1 octet	Meaning
00-14-72	0	Reserved
00-14-72	1	WAI Certificate Authentication and Key Management
00-14-72	2	WAI Preshared Key Authentication and Key Management
00-14-72	3-255	Reserved
Others	0-255	Reserved

7.3.2.9.2 Unicast Cipher Suites

The suite selector format is shown in Figure 41d.

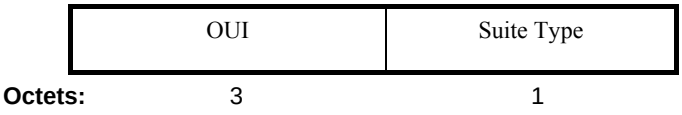

Figure 41d—Suite selector format

The suites defined in this specification are shown in Table 19b.

Table 19b Cipher Suite

OUI 3 octets	Type 1 octet	Meaning
00-14-72	0	Reserved
00-14-72	1	WPI-SMS4
00-14-72	2~255	Reserved
others	0~255	Reserved

7.3.2.9.3 Multicast Cipher Suites

The suites defined in this specification are shown in Table 19c.

Table 19c Cipher Suite

OUI 3 octets	Type 1 octet	Meaning
00-14-72	0	Reserved
00-14-72	1	WPI-SMS4
00-14-72	2~255	Reserved
others	0~255	Reserved

8 Security

The WAPI mechanism is adopted to implement security access control and data confidentiality. The WAPI security is made up of two parts:

— WAI authentication and key management.

— WPI data transport protection.

8.1 WAI authentication and key management

The WAI authentication and key management is comprised of the four mechanisms described below:

a) Based on a certificate in a BSS

1) The STA identifies the WAI authentication and key management suites supported by the AP through passively monitoring Beacon frames or through actively probing (Probe Response frames).

2) Authentication is executed between the STA and the AP.

3) During (re)association, the STA determines the cipher suite based on the WAPI parameter in the

(Re)Association request frame.

4) The STA and AP execute an authentication process to negotiation a BK.
5) The STA and AP execute the unicast key negotiation and multicast key announcement processes.
6) The WPI module is informed of the negotiated keys and cipher suites, and then uses them to protect the data.

b) Based on a PSK(between the STA and AP)in a BSS
   1) The STA identifies the WAI authentication and key management suites by passively monitoring Beacon frame or through actively probing (Probe Response frame).
   2) Authentication is executed between the STA and AP.
   3) During (re)association, the STA determines the cipher suite based on the WAPI parameter in the (Re) Association request frame.
   4) The STA and an AP will execute the unicast key negotiation and multicast key announcement process after the BK is derived from a PSK.
   5) The WPI module is informed with the negotiated keys and cipher suites, and then uses them to protect the data.

c) Based on a certificate in an IBSS
   1) The STA identifies the WAI authentication and key management suite supported by the peer STA passively monitoring Beacon frame or through actively probing (Probe Response frame).
   2) An optional authentication is executed between the STA and the peer STA.
   3) The STA and the peer STA execute the certificate authentication process to negotiate a BK.
   4) The STA and the peer STA execute the unicast key negotiation and the multicast key announcement process.
   5) The WPI module is informed with the negotiated keys and cipher suites, and then uses them to protect the data.

d) Based on a PSK in an IBSS
   1) The STA identifies the WAI authentication and key management suite by passively monitoring Beacon frame or through actively probing (Probe Response frame).
   2) An optional authentication is executed between the STA and the peer STA.
   3) The STA and the peer STA will execute the unicast key negotiation and the multicast key announcement process after the BK is derived from a PSK.
   4) The WPI module is informed with the negotiated keys and cipher suites, and then uses them to protect the data.

If the WAPI security mechanism is negotiated to enable during (re)association between the STA and AP /STA, the mutual authentication and key negotiation must be executed. Based on a certificate, the entire authentication procedure should include the certificate authentication, the unicast key negotiation and the multicast key announcement. Based on a PSK, the entire authentication procedure should include the unicast key negotiation and the multicast key announcement.

The authentication messages between the STA and AP /STA shall be transported by the WAPI protocol with Ether Type 0x88B4, and those packets between the AP/STA and Authenticator Server are transported by the UDP socket with port number 3810.

A WAPI security association starts when the STA’s SME decides to establish a WAPI security network and

completes when the MLME-SETPROTECTION.Request primitive is invoked. The time that the security association takes to set up must be less than the MIB variable dot11WAPIConfigSATimeout. If the time is greater than the MIB variable dot11WAPIConfigSATimeout, the Deauthentication frame will be sent.

8.1.1 The structure of the authentication system

8.1.1.1 Systems and Ports

The STA provides two kinds of logical channels to access LAN. They are defined as two kinds of ports, the controlled port and the uncontrolled port.

An STA or an AP provides a port, which is an uncontrolled port, for other STAs to connect to the Authenticator Server. The STA or the AP also provides the controlled port. The controlled port shall block data transmission and reception until after a successful authentication

8.1.1.2 Controlled and Uncontrolled access

The access control of an authentication subsystem is based on the Port， shown as Figure 43a.
The uncontrolled port allows the transmission of authentication data (WAPI protocol data) in WLAN, this transmission process is unrestricted by the current authentication state. The general protocol data (except WAPI protocol data) will pass through the controlled port only if the authentication state of this port is authenticated. The controlled and the uncontrolled port may be two logical ports that are connected to the same physical port. All the data through the physical port can reach both of the ports. The flow of data will be determined by the current authentication state.

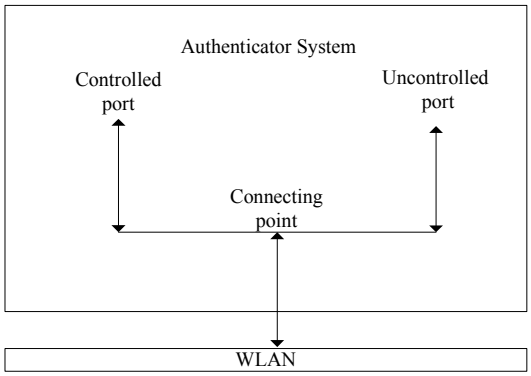

Figure 43a—Uncontrolled and controlled ports

Figure 43b depicts two different authentication states associated with the controlled port, representing the state as a switch that can be turned on or off. When the ‘switch’ is turned off, the controlled Port is unauthorized and is therefore disabled and prevents the flow of PDUs; when the ‘switch’ is turned on, the controlled port is authorized and is therefore enabled and allows the flow of PDUs. There are two subsystems in Figure 43b. In authentication subsystem 1, the authentication state of the controlled port is unauthenticated and no data is permitted to across this port. In the subsystem 2, the state is authenticated and PDUs are permitted to across the port.

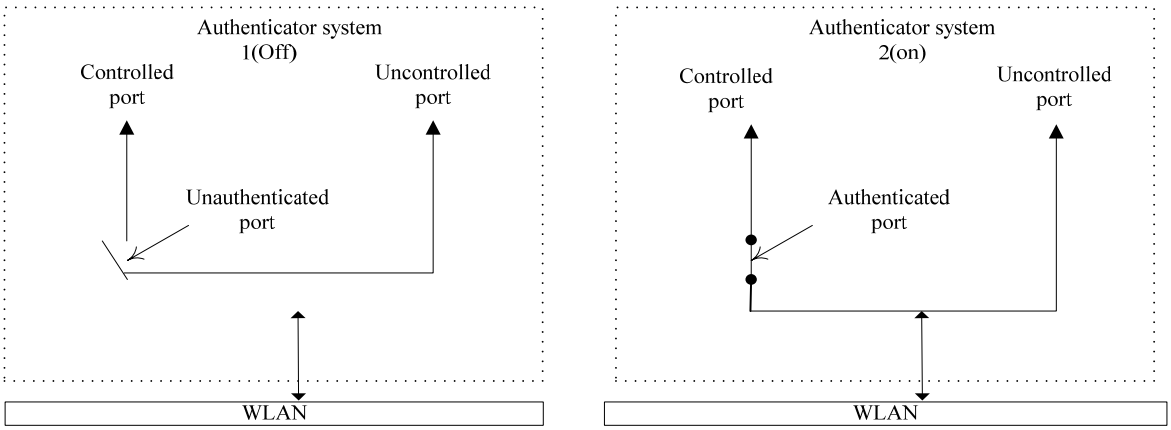

Figure 43b—Authentication state on controlled port

The state of any controlled port of the system is determined by the authentication control parameter of the system. The value of the system control parameter may be “authentication disabled” or “authentication enabled”. When the system control parameter is set to “authentication disabled”, the state of all the controlled ports is “authenticated”. When it is set to “authentication enabled”, the authentication state of every controlled port of the system is determined by the following two types of authentication control style:

a) ForceUnauthenticated: the state of a certain controlled port is forced to be unauthenticated. That is, the state of a controlled port is unconditionally set to unauthenticated (i.e. no data is permitted to transit this controlled port)

b) Auto: the state of a controlled port is set according to the result of the mutual authentication between the Authenticator and the Supplicant (i.e. the data can pass through the controlled port only if the authentication is successful.).

In the system, the networking protocol data except the authentication data between STAs is transmitted out through the controlled ports. Figure 43c gives the logical structure of the controlled port and the uncontrolled port. The authentication state of the controlled port in the system is set by Authenticator according to the result of the authentication exchange.

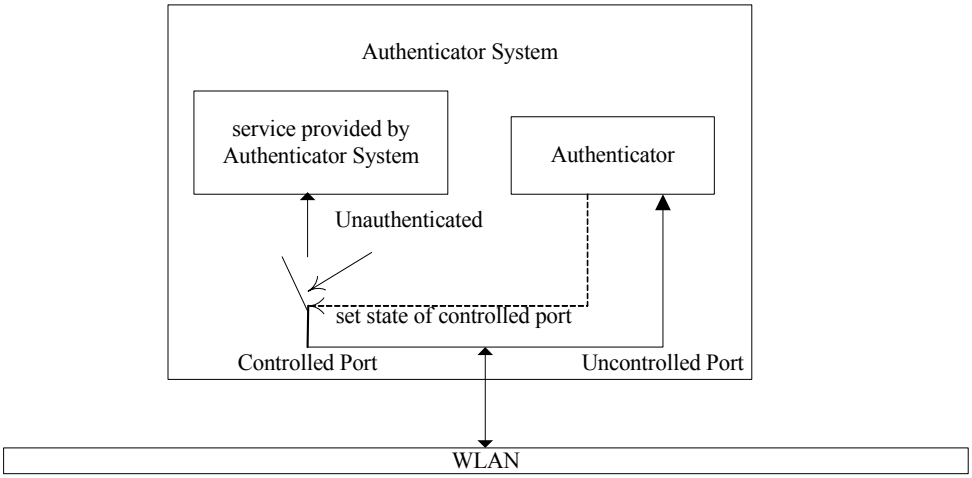

Figure 43c—Usage of the controlled and uncontrolled port

Figure 43d shows the relations and the information exchange process among Supplicant, Authenticator and Authenticator Server. In this figure, the state of the controlled port in the Authenticator is unauthenticated, which causes the Authenticator system to deny the data passing from the controlled port.

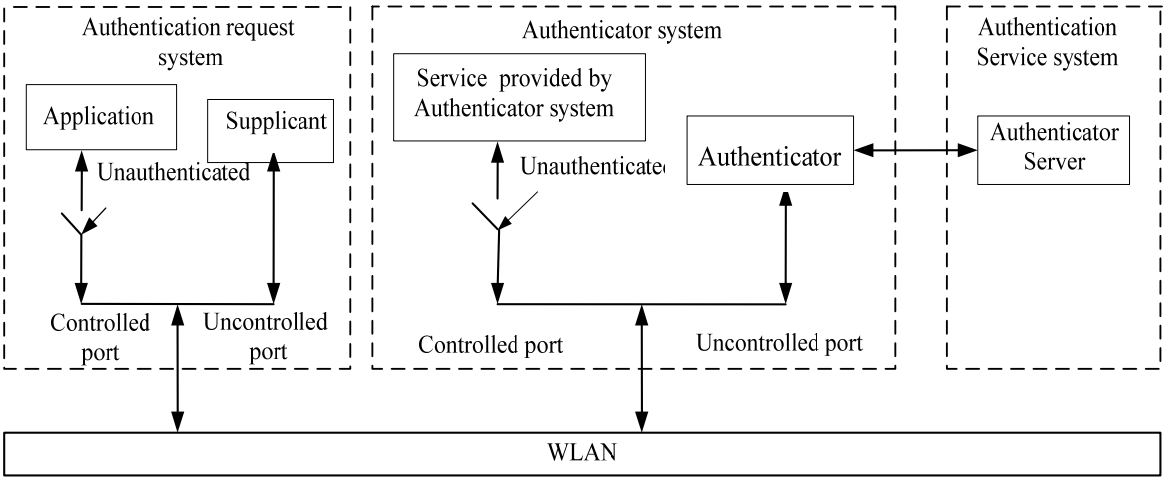

Figure 43d—Roles of Supplicant, Authenticator and Authenticator Server

8.1.2 WAPI security association management

8.1.2.1 WAPI security association definitions

A security association is a set of policy(ies) and key(s) used to protect information. There are 4 kinds of security associations in WAPI:

— BKSA: The base key security association is from the PSK information or the result of a successful WAI certificate authentication exchange.

— USKSA: The unicast session key security association is the result of a successful unicast key negotiation handshake.

— MSKSA: The multicast session key security association is the result of a successful multicast key announcement handshake.

— STAKeySA: STAKey security association is the result of a successful STAKey announcement handshake.

BKSA

The BKSA is bidirectional. After a successful WAI certificate authentication or PSK configuration, BKSAs are established in the Supplicant and Authenticator and held for a period not exceeding the life cycle of each. The BKSA is used to create the USKSA. A BKSA consists of the following elements:

— BKID: the identifier of BKSA

—Authenticator MAC address
—Supplicant MAC address
— BK
— Lifetime
— AKMP
— Other security parameters（Optional）

USKSA

The USKSA, which is the result of a successful unicast key negotiation, is also bidirectional. With its Negotiation based on a BK, the USKSA is independent of its superordinate BKSA and held for its lifetime. For each Supplicant and Authenticator pair, there are two USKSAs at most. Generally, only one of the USKSAs is active in a BSS. However, both of them may be active during rekeying process. After a unicast data MPDU encrypted with a new correctly received USKSA , the old key is set to invalid. In an IBSS, if $MAC_{Authenticator}$（the MAC address of the Authenticator）> $MAC_{Supplicant}$ （the MAC address of the Supplicant）, the negotiated USKSA is used on unicast data MPDU and the key updating state is the same as in a BSS. If $MAC_{Authenticator} < MAC_{Supplicant}$, the negotiated USKSA can only be used in the multicast key announcement. When a new USKSA is active, the old one is set to invalid immediately. The USKSA consists of the following elements:

— USKID
— USK
— The selected unicast cipher suite
— Lifetime
—Authenticator MAC address
—Supplicant MAC address
— Other security parameters, such as replay counter used in pre-authentication and STAKey request.

MSKSA

The MSKSA is the result of a successful multicast key announcement. In a BSS, only one MSKSA is active. It is used to encrypt broadcast/multicast MPDUs by the AP or decrypt received broadcast/multicast MPDUs by the STA. In an IBSS, every STA has more than one active MSKSAs, one for sending encrypted broadcast/multicast MPDUs and others for decrypting broadcast/multicast MPDUs sent by corresponding peer STAs. During the MSKSA refreshing, for every AP or peer STA, a STA has two active MSKSAs used to decrypt broadcast/multicast MPDUs. Only after a broadcast/multicast MPDU is correctly decrypted with the new MSKSA, the old ones are set to invalid. However, for the MSKSA used to encrypt broadcast/multicast MPDUs, only one is active. The MSKSA consists of the following elements:

— Direction vector (whether the MSK is used for transmit or receive)
— MSKID
— The selected Multicast cipher suite
— Lifetime
— MSK
— Authenticator MAC address
— Other security parameters（Optional）

STAKeySA

The STAKeySA is the result of a successful STAKey negotiation. It is an unidirectional security association from the initiator STA to the peer STA. The MSKSA consists of the following elements:

— STAKeyID
— STAKey
—Unicast cipher suite (adopts multicast cipher suite announced by the AP)
— Initiator STA MAC address
— Peer STA MAC address

8.1.2.1.1 Security association in an ESS

When WAPI is enabled，the STA/AP is identified by the WAPI parameter set which is contained in the Beacon frame or the Probe Response frame. If there is no such information contained in the frame, the WAPI security mechanism is not adopted in that STA/AP.

There are three ways to establish BKSA for a roaming STA in an ESS:

— After association, execute the certificate authentication or directly establish the BKSA by the PSK.
— After a certificate authentication with an AP in the ESS, the STA can store that BKSA. When the STA moves to another AP, the STA may place BKIDs corresponding to one or more BKSAs in the WAPI security parameter set of the (re) association request packet. If there is a valid corresponding BKSA in the AP, then the unicast key negotiation process can be directly executed without certificate authentication. If there is no valid corresponding BKSA in the AP or there is no BKID in the WAPI security parameter set of the (re) association request packet of the STA, the BKSA must be established by the PSK or a certificate authentication between the STA and the AP.
— If a STA has completed the establishment of BKSA and USKSA with an AP in an ESS. It may establish a BKSA with other APs in the ESS through a pre-authentication before association. The STA and the AP store the BKSA after a successful pre-authentication.

Based on BKSA, USKSA can be established with the unicast key negotiation process and MSKSA with the multicast key announcement process.

8.1.2.1.2 Security association in an IBSS

If WAPI is enabled in an IBSS, a security association should be established between each pair of STAs. A pair of STAs may negotiate a unicast cipher suite supported by both of them. Every STA shall include its list of the unicast cipher suite and the multicast cipher suite in its Beacon frame or Probe Response frame. The two STAs must support a common subset of unicast cipher suites and each STA shall support the multicast cipher suite broadcasted by the peer STA, otherwise BKSA between them can not be established.

In an IBSS, if a STA requires establishing a security association with another STA whose security policy is unknown to it. It must send a Probe Request frame to obtain the security policy of the peer STA. The two STAs negotiate unicast cipher algorithms through the unicast key negotiation process. Selected unicast cipher suite is contained in the WAPI parameter set of the unicast key negotiation response. However, the WAPI parameter set contained in the unicast key negotiation acknowledgement is the same as in the Beacon frame or the Probe

Response frame. Two STAs will use the unicast cipher algorithm through the unicast key negotiation initiated by the STA (as Authenticator) with the larger MAC address.

If the certificate-based authentication is negotiated between two STAs, each of them initiates its own certificate authentication, unicast key negotiation process and multicast key announcement process. Two sets of BKSAs, USKSAs and MSKSAs are established. If the PSK-based authentication is negotiated，a BK is derived from the PSK, a BKSA is established. Based on the BKSA, each of them initiates their own unicast key negotiation process and multicast key announcement process to establish two USKSAs and two MSKSAs.

8.1.2.1.3 Deletion of the WAPI security association

When the STA/AP receives the primitive of Association, (Re)Association, Disassociation, Authentication, De authentication, or it believes that it has drifted out of radio range of a STA/AP, some security associations will be deleted.

In a BSS, a non-AP STA will delete the USKSA and MSKSA, and the AP will delete the USKSA.

In an IBSS, the STA will delete the USKSA and the MSKSA used to receive the data.

If the lifetime of a security association is run out or the state of a security association is set to invalid, the security association will be deleted. Based on its own management policy, a STA may set the state of some security associations to invalid.

8.1.2.2 Selection of the WAPI security policy

The selection of the WAPI mechanism is decided by the process of association. the STA executes the WAPI policy selection by setting the WAPI parameter set in the (re-)association frame.

The STA/AP identifies its supporting WAPI security policy by the WAPI parameter set contained in the Beacon frame or in the Probe Response frame. From the supporting WAPI security policy, the other STAs that require to associate with it should choose the proper WAPI security policy. The association STA identifies the chosen WAPI security policy with the WAPI parameter set field contained in the association request frame. Thus, the selection of security policy is decided. If the association STA does not support the notified policy, the association request is denied.

The STA and AP establish secure network by the steps below:

a) the STA chooses the AP by the SSID
b) the STA and AP execute Authentication
c) the STA and AP execute association process to decide the selection of WAPI security mechanism. Based on the selection, the STA and AP execute the WAI protocol described in 8.1.4.

In an IBSS, the STA chooses the WAPI security policy from the WAPI parameter set contained in the Beacon frame or in the Probe Response frame sent by the peer STA. The policy includes the way of authentication, key management and multicast cipher suite. The unicast cipher suite which is negotiated in the process of the unicast key negotiation is based on the authentication and key management method in the chosen WAPI security policy.

8.1.3 Certificate

Authenticator Server is an important component of WAI authentication infrastructure based on public-key cryptography technology. Its fundamental function is to verify the certificate of the user.

User certificate, which is a public-key certificate, is an important part of the construction of WAI system. The identity of the user can be uniquely identified by the certificate and the corresponding secret key. Public-key certificate is the digital identification of user.

Two kinds of public-key certificate format, X.509 v3 and GBW, are supported in this specification. User certificate and Authenticator Server certificate are used for signature verification.

8.1.3.1 X.509 v3 Certificate

The format and management of X.509 v3 certificate is defined in ITU-T Recommendation X.509，Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks.

The signature algorithm in certificate is 192-bit ECDSA. Hash algorithm is SHA256. The ECDSA algorithm is defined in ANSI X9.62-Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm(ECDSA). The SHA256 algorithm is defined in FIPS 180-2, Secure Hash Standard (SHS), August 2002.
The parameter for the ECDSA is denoted by OID.

Note: One set of appropriate parameters in ECDSA shall be adopted here, and the concrete values of parameters to be adopted are optional.

8.1.3.2 GBW Certificate

8.1.3.2.1 Certificate description

Version
Serial number
Issuer name
Validity period
Subject name
Subject public key information

Extension
Signature algorithm Signature of the certificate by the issuer

Figure 43e—Format of certificate

Version

This field determines the format of the certificate, which makes corresponding protocol be able to extract useful data items from the certificate.

Serial number

Each certificate issued by Authenticator Server is assigned an unique serial number. Along with the issuer, it can uniquely determine the certificate subject who possesses the certificate.

Issuer name

This field designates the identity of the issuer.

Validity period

This field prescribes the time interval during which the certificate is valid. It adopts UTC time format, which denotes the number of seconds from 0 o'clock on 1970.1.1 to current time.

Subject name

This field specifies the identity of who possesses the certificate.

Subject public key information

This field is the public-key information of the subject.

Extension

This field is kept for extension applications in the future.

Signature algorithm

This field specifies the signature algorithm used by the issuer on Signature of the certificate. Along with the public key contained in the certificate, it can be used to verify a signature of the subject. It should include a hash algorithm a digital signature algorithm and corresponding parameters.

Signature of the certificate by the issuer

This field is the certificate issuer (Authenticator Server)'s signature of all the fields excluding itself.

8.1.3.2.2 Certificate format

The format of GBW certificate is as below:

Version	Serial number	Issuer	Validity period	Subject	Subject public key	Extension	Extension Attributes	Signature algorithm	Signature on certificate

Octets:	2	4	6-256	8	6-256	variable	1	variable	2	variable

Figure 43f—Definition of certificate content

— The Version field has the length of 2 octets, and the current version is 1.

— The Serial number field has the length of 4 octets.

— The issuer field contains a length field and a content field. The length field denotes the length of the issuer field in octet and the length of itself is 1 octet.

— The Validity period field has the length of 8 octets, of which 4 is for beginning time and 4 for ending time. Both the beginning time and the ending time are in the UTC format.

— The Subject field contains a length field and a subfield. The length field denotes the length of the subject field in octet and the length of itself is 1 octet.

— The Subject public-key field is composed of two fields that are the length field and the content field. The length field denotes the length of subject public-key field, with the length of 2 octets. The content field contains three subfields that are public-key algorithm identifier, parameter and public key.

a) The Public-key Algorithm Identifier subfield is 1 octet in length, where the value 1 stands for ECDSA-192 and other values are reserved.

b) The Parameter subfield is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The Length field is 2 octets in length and denotes the number of octets in the Content field.

The value of Parameter Identifier field is defined as follow when the value of the Public-key Algorithm Identifier field is 1:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters q, a, b, h, n, and G. A set of EC domain parameters specifies a field $GF(q)$, where q is a positive odd prime integer p, 2^m for some positive integer m, or p^m for an odd prime p and some integer $m \geq 2$; two elliptic curve coefficients a and b, elements of $GF(q)$, that define an elliptic curve E; a positive prime integer n dividing the number of points on E; and a curve point G of order n (G is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on E). The order of the detailed parameters in the field is q, a, b, h, n, and G. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

c) The Public key Algorithm Identifier subfield denotes the public key value. It has the length of 48 octets when the public key identifier is 1.

— The Extension field has the length of 1 octet. It gives the number of the Extension Attributes.

— The Extension Attributes field contains the certificate extensions. Every extension is defined as TLV format.

— The Signature algorithm field shows which algorithm should be used on signature of the certificate by the issuer. It contains two subfields that are Length and Content. The Length subfield denotes the length of the Content field in octet with the Length of 2 octets. The Content subfield is composed of an one-octet hash algorithm identifier, an one-octet signature algorithm identifier and a parameter.

a) Hash algorithm identifier is defined as below:

1 denotes SHA-256.

Other values are reserved.

b) Signature algorithm identifier

1 denotes ECDSA-192.

Other values are reserved.

c) Parameter denotes the parameter information of the signature algorithm. It is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The Length field is 2 octets in length and denotes the octet number of Content field.

The value of parameter Identifier field is defined as follow when the value of Signature algorithm identifier is 1:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters q, a, b, h, n, and G. A set of EC domain parameters specifies a field $GF(q)$, where q is a positive odd prime integer p, 2^m for some positive integer m, or p^m for an odd prime p and some integer $m \geq 2$; two elliptic curve coefficients a and b, elements of $GF(q)$, that define an elliptic curve E; a positive prime integer n dividing the number of points on E; and a curve point G of order n (G is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on E). The order of the detailed parameters in the field is q, a, b, h, n, and G. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

— Signature on certificate field contains Length and Value fields. The Length field denotes the length of the Value field, with the length of 2 octets. Value field is the signature value of all the fields described above (except Signature on certificate field) by the issuer.

The management method of the certificate issuing and revoking, as well as the communication between two Authenticator Servers, is beyond this specification.

The Extension Attribute adopts the Type-Length-Value (TLV) format as below:

	Type	Length	Value
Octets:	1	2	variable

Figure 43g—Extension Attribute format

The 1-octet Type field denotes the type of this attribute.

1 Authority Key Identifier

Other values are reserved

The 2-octet Length field gives the number of octets in the Value field.

The Value field contains the content of this attribute data.

Authority Key Identifier attribute provides a method of identifying the public key corresponding to the private key used to sign a certificate. The Value field of Authority Key Identifier attribute contains an Issuer Name subfield and a Serial Number subfield of the issuer certificate. Here, the Issuer Name field consists of a Length subfield and an Issuer Name Content field and the 2-octet Length field gives the number of octets in Issuer Name Content field; the Serial Number field consists of a Length field and a Serial Number Content field and the 2-octet length field gives the number of octets in Serial Number Content field.

8.1.3.2.3 Issue format of the certificate

The default suffix of the issue certificate file is .wcr. It is stored on big endian mode which is network octet order.

The format of the issue certificate is as below:

	Certificate file identification	Version	Certificate file length	Digest	Attribute1	Attribute2	Attribute3
Octets:	16	2	2	4-65539	4-65539	4-65539	4-65539

Figure 43h—Issue format of the certificate

The certificate file identification field shows the storage identification of the GBW certificate file. It is used to check whether the file is an issued certificate file of GBW type. If so, the value of this field should be "WAI15629.11-2003" with the form of ASCII code character string (denoted in hex as 0x57 0x41 0x49 0x31 0x35 0x36 0x32 0x39 0x2E 0x31 0x31 0x2D 0x32 0x30 0x30 0x33, with the length of 16 octets.)

The version field is the version of the storage format of the certificate file, and its current value is 1, with the length of 2 octets.

The certificate file length field indicates the length of the digest and all attribute fields in octet, with the length of 2 octets.

The digest field is the digest of all the fields behind the digest field by certain digest algorithm. The format of the digest field is as below:

	Digest algorithm identification	Digest length	Digest data
Octets:	2	2	0-65535

Figure 43i—Digest field

The Digest algorithm identification field identifies the digest algorithm, with the length of 2 octets. Digest algorithm identification is defined as below:

— 1 denotes SHA-256 hash algorithm. If the value is 1, the digest length value is 36 and digest data is 32-octet digest content.

— Other values are reserved.

The Digest length field denotes the length of the digest data field in octet, with the length of 2 octets.

The Digest data field is the content of the digest.

There may be several attribute fields in the certificate file. The format of the attribute field is as below:

	Attribute identification	Attribute length	Attribute data
Octets:	2	2	0-65531

Figure 43j—Attribute field

The Attribute identification field denotes the type of attribute, with the length of 2 octets. It is defined as below:

— 1 the attribute data is the certificate of the issuer
— 2 the attribute data is the certificate of the user
— 3 the attribute data is the secret key corresponding to user certificate.
— Other values are reserved

The attribute length field shows the length of the attribute data in octet, with the length of 2 octets.

The Attribute data field is the content of the attribute.

If a private key attribute according to the user certificate is contained in the certificate file, the last attribute field before this attribute field shall be the certificate according to this private key.

8.1.4 WAI protocol

8.1.4.1 WAI protocol packet format

The octets in the fields longer than a single octet are sent by the order from the octet containing the highest numbered bits to the octet containing the lowest numbered bits (Network Byte Order).

The format of the WAI protocol packet is defined as below:

	Version	Type	Sub type	Reserved	Length	Packet sequence number	Fragment sequence number	Flag	Data
Octets:	2	1	1	2	2	2	1	1	variable

Figure 43k—Format of WAPI protocol packet in WAI authentication system

— The Version field denotes the version of authentication infrastructure, with the length of 2 octets. Current version value is 1.

— The Type field denotes the protocol type, with the length of 1 octet. It is defined as below:

1 denotes WAI protocol package.

Other values are reserved.

— The Subtype field is 1 octet in length. If the value of the Type field is not 1, the Subtype field is reserved. Otherwise, the Subtype field is defined as below:

1 pre-authentication start
2 STAKey request
3 authentication activation
4 access authentication request
5 access authentication response
6 certificate authentication request
7 certificate authentication response
8 unicast key negotiation request
9 unicast key negotiation response
10 unicast key negotiation confirmation
11 multicast key/STAKey announcement
12 multicast key/STAKey announcement response

Other values are reserved.

— The Reserved field is 2 octets in length, with the default value 0.

— The Length field denotes the length of all the fields of the WAPI protocol packet in octet, with the length of 2 octets.

— The Packet Sequence Number field denotes the sequence number of the packet, with the length of 2 octets.

— The Fragment Sequence Number denotes the sequence number of the fragment, with the length of 1 octet. For every packet, the sequence number of the first fragment is 0, and that of the following is increased by degree of 1.

— The Flag field is 1 octet in length in which bit 0 is More Frag field, bits 1-7 are reserved. The More Frag field is set to 1 in packet that has another fragment of the current WAI packet to follow. It is set to 0 in all other frames.

The Packet Sequence Number, Fragment Sequence Number and More Frag field are valid only in those packets between Supplicant and Authenticator.

— The content of the Data field is determined by the values of the Type field and the Subtype field. Besides the fixed content, some optional attributes are also included in this field.

The maximal length of the WAI protocol packet is 65535 octets.

8.1.4.1.1 Fixed content in the data field of the WAI protocol packet

a) FLAG

This field is 1 octet in length, with the format below:

B0	B1	B2	B3	B4	B5	B6	B7

BK Rekeying	Pre-Authentication	Certificate Authentication Request	Optional Field	USK Rekeying	STAKey Negotiation	STAKey Revoking	Reserved

Figure 43l—FLAG

— Bit 0: BK Rekeying: A STA sets BK Rekeying subfield to 1 to signal the packet is a BK rekeying packet and set this subfield to 0 when the packet is not a BK rekeying packet.

— Bit 1: Pre-Authentication: A STA sets Pre-Authentication subfield to 1 to signal the packet is a pre-authentication packet and set this subfield to 0 when the packet is not a pre-authentication packet.

— Bit 2: Certificate Authentication Request: A STA sets this subfield to 1 if it requires to verify the peer's certificate, otherwise sets the subfield to 0.

— Bit 3: Optional Field : If there are some optional fields in the packet, this subfield is set to 1. Otherwise this subfield is set to 0.

— Bit 4: USK Rekeying: A STA sets USK Rekeying subfield to 1 to signal the packet is a USK rekeying packet and set this subfield to 0 when the packet is not a USK rekeying packet.

— Bit 5: STAKey Negotiation: A STA sets STAKey Negotiation subfield to 1 to signal the packet is a STAKey negotiation packet and set this subfield to 0 when the packet is not a STAKey negotiation packet.

— Bit 6: STAKey Revoking: This subfield is set to 1 when a STA delete a STAKey and set to 0 otherwise.

— Bit 7: Reserved.

b) BKID

BK identifier, 16 octets in length, is computed as BKID= KD_HMAC_SHA256(BK, $MAC_{Authenticator}$ ||$MAC_{Supplicant}$) .

c) USKID

USKID is 1 octet in length, in which bit 0 is significant.

d) MSKID/STAKeyID

MSKID / STAKeyID is 1 octet in length, in which bit 0 is significant.

e) Result

This field is 1 octet in length. The value may be 0 (success) or other values (reason code).

f) Nonce

This field is 32 octets in length.

g) Key Data

This field is composed of two subfields which are the length subfield and the content subfield. The length subfield is 1 octet in length and denotes the number of octets in the content field.

h) Key Announcement Identifier

This field is 16 octets in length.

i) Data Sequence Number

This field is 16 octets in length.

j) Certificate

	Certificate Identifier	Certificate Length	Certificate Data
Octets:	**2**	**2**	**0-65535**

Figure 43m—Certificate

— The Certificate Identifier subfield denotes the type of the certificate, with the length of 2 octets. It is defined as below:

1 denotes X.509 v3 certificate in the Certificate Data subfield.

2 denotes GBW certificate in the Certificate Data subfield

Other values are reserved.

— The Certificate Length subfield gives the number of octets in the Certificate Data subfield, with the length of 2 octets.

— The Certificate Data subfield contains the certificate content.

k) Identity

	Identity Identifier	Identity Length	Identity Data
Octets:	**2**	**2**	**0-65535**

Figure 43n—Identity

— The Identity Identifier subfield denotes the type of Identity, with the length of 2 octets. It is defined as below:

1 The Identity Data subfield contains three subfields: the Subject subfield, the Issuer subfield and the Serial Number subfield.

Other values are reserved.

— The Identity Length subfield gives the number of octets in the Identity Data subfield, with the length of 2 octets.

— The Identity Data subfield contains the identity content. When the Identity Identifier subfield is set to 1, the Subject subfield contains a length field and a subject content field. The 2-octet length field gives the number of octets in subject content field; the Issuer subfield contains a length field and an issuer content field. The 2-octet length field gives the number of octets in issuer content field; the Serial Number subfield contains a length field and a serial number content field. The 2-octet length field gives the number of octets in serial number content field.

l) ADDID

This field is 12 octets in length.

	MAC address 1	MAC address 2
Octets:	**6**	**6**

Figure 43o—ADDID

8.1.4.1.2 Attribute content in the data field of the WAI protocol packet

The attribute content adopts the Type-Length-Value (TLV) format as below:

	Type	Length	Value
Octets:	1	2	variable

Figure 43p—Attribute format

The 1-octet Type field denotes the type of this attribute.

The 2-octet Length field gives the number of octets in the Value field.

The Value field contains the content of this attribute.

a) Signature attribute

	Type=1	Length	Identity	Signature Algorithm	Signature Value
Octets:	1	2	variable	variable	variable

Figure 43q—Signature attribute

The Identity field has the same definition as 8.1.4.1.1 (k).

The Signature Algorithm field contains two subfields that are the length subfield and the content subfield. The length subfield is 2 octets in length and gives the length of the content subfield in octet. The content subfield contains three subfields: a one-octet hash algorithm identifier, a one-octet signature algorithm identifier and parameter.

1) Hash algorithm identifier is defined as below:
   1 SHA-256.
   Other values are reserved.
2) Signature algorithm identifier
   1 ECDSA-192.
   Other values are reserved.
3) Parameter contains the parameter information of the signature algorithm. It is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The length field is 2 octets in length and gives the number of octet in Content field.

The value of Parameter Identifier is defined as follow when the value of Signature Algorithm Identifier is 1:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters q, a, b, h, n, and G. A set of EC domain parameters specifies a field $GF(q)$, where q is a positive odd prime integer p, 2^m for some positive integer m, or p^m for an odd prime p and some integer $m \geq 2$; two elliptic curve coefficients a and b, elements of $GF(q)$, that define an elliptic curve E; a positive prime integer n dividing the number of points on E; and a curve point G of order n (G is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on E). The order of the detailed parameters in the field is q, a, b, h, n, and G. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

The Signature Value field contains a Length subfield and a Content subfield. The Length field is 2 octets in length and gives the number of octets in the Content field. The Content subfield contains the signature value.

b) Certificate Verification Result

type=2 (1 octet)	length (2 octets)
Nonce 1 (32 octets)	
Nonce 2 (32 octets)	
verification result (1 octet)	Certificate1 (variable)
verification result (1 octet)	Certificate2 (variable)

Figure 43r—Certificate Verification Result

c) Identity List

type=3 (1 octet)	length (2 octets)
reserved (1 octet)	number of Identities (2 octets)
Identity1 (variable)	
Identity2 (variable)	
…	

Figure 43s—Identity List

The definition of Identity 1, Identity 2, Identity3, … should be the same definition as 8.1.4.1.1(k).

8.1.4.2 Certificate Authentication procedure

The Certificate Authentication procedure executes the authentication and the key negotiation to establish BKSA based on the certificate of the STA/AP. It is depicted by Figure 43t.

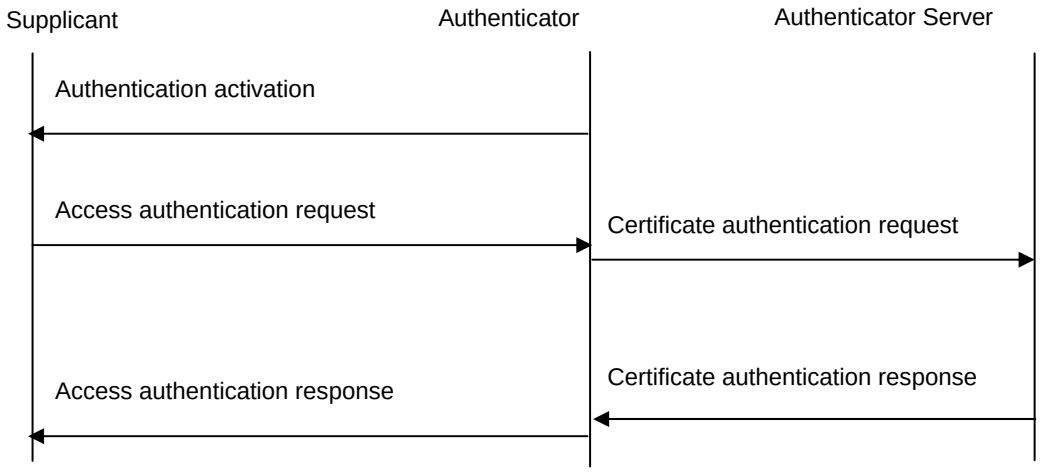

Figure 43t—Certificate Authentication procedure

8.1.4.2.1 Authentication Activation

The authentication activation packet is illustrated in Figure 43u.

	FLAG	Authentication Identifier	Local Authenticator Server Identity	$STA_{Authenticator}$ Certificate	ECDH Parameter	Signature
Octets:	1	32	variable	variable	variable	variable

Figure 43u—The format of the Data field of Authentication Activation packet

— The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Only bit 0 and bit 1 are significant here. When the certificate authentication procedure is for the initial BKSA establishment, bit 0 (BK Rekeying) is set to 0. When the certificate authentication procedure is for BK rekeying, bit 0(BK Rekeying) is set to 1. If the certificate authentication procedure is for Pre-Authentication, bit 1 (Pre-Authentication) is set to 1; otherwise, bit 1 is set to 0.

—The Authentication Identifier field is 32 octets in length. If the BK Rekeying is 0, the Authentication Identifier field is a random number generated by Authenticator. If the BK Rekeying is 1, the Authentication Identifier field is equal to the value negotiated in the last certificate authentication procedure.

— The Local Authenticator Server Identity field uses the same definition as that in 8.1.4.1.1. It describes the Authenticator Server that the Authenticator trusts.

— The $STA_{Authenticator}$ Certificate field contains the certificate of $STA_{Authenticator}$, whose format is the same definition as that in 8.1.4.1.1.

— The ECDH Parameter field is 1 octet in length and indicates the parameter used by ECDH. It is composed of Parameter Identifier, Length and Content fields. The Parameter Identifier field is 1 octet in length. The Length field is 2 octets in length and gives the number of octets in the Content field.

The value of Parameter Identifier field is as follow:

1 The Content field is OID which describes the parameter set. The code rule of OID is based on DER.

2 The Content field is the detailed parameters value. It defines the detailed parameters q, a, b, h, n, and G. A set of EC domain parameters specifies a field $GF(q)$, where q is a positive odd prime integer p, 2^m for some positive integer m, or p^m for an odd prime p and some integer $m \geq 2$; two elliptic curve coefficients a and b, elements of $GF(q)$, that define an elliptic curve E; a positive prime integer n dividing the number of points on E; and a curve point G of order n (G is called the *generator* of a subgroup of order n). The cofactor $h = \#E / r$ (where $\#E$ is the number of points on E). The order of the detailed parameters in the field is q, a, b, h, n, and G. Each parameter is described by length and value. The Length subfield is 1 octet in length and gives the number of octets in value subfield.

Other Parameter Identifiers are reserved.

The ECDH algorithm is defined in ISO/IEC 15946 Information technology - Security techniques - Cryptographic techniques based on elliptic curves.

The Authenticator sends an authentication activation packet to Supplicant when:

a) certificate authentication and key management method is chosen in (re)association
b) re-authentication and rekeying is required according to the local policy of the Authenticator
c) the Authenticator receives a Pre-authentication Start packet from the Supplicant.

After the Supplicant receives the Authentication Activation packet from the Authenticator,

a) If the value of bit 0 (BK Rekeying) of the FLAG field in the packet is 1, the Supplicant proceeds to Step b).Otherwise, goes to Step c).
b) If the Authentication Identifier of the packet dose not match the one kept in the Supplicant which is negotiated in the last Certificate Authentication procedure, the Supplicant silently discards the packet. Otherwise the Supplicant proceeds to Step c).
c) The Supplicant chooses the corresponding $STA_{Supplicant}$ certificate according to the Authenticator Server Identity in the Authentication Activation packet or local strategies. And it generates temporary secret key x, temporary public key x·P for ECDH exchange and a Supplicant challenge. The Supplicant constructs the Access Authentication Request packet and sends it to the Authenticator.

8.1.4.2.2 Access Authentication Request

The Access Authentication Request packet is illustrated in Figure 43v.

	FLAG	Authentication Identifier	Supplicant Challenge	Supplicant Key Data	$STA_{Authenticator}$ Identity	$STA_{Supplicant}$ Certificate	ECDH Parameter	Authenticator Server List trusted by Supplicant	Supplicant Signature
Octets:	**1**	**32**	**32**	**variable**	**variable**	**variable**	**variable**	**variable**	**variable**

Figure 43v—The format of the Data field of access authentication request packet

— The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here only bits 0, 1, 2 and 3 are significant. Bit 0 and Bit 1 should have the same value as the FLAG filed in the Authentication Activation. The Supplicant sets Bit 2 (Certificate Verification Request Identifier) to 1 if it requires to verify the Authenticator certificate; otherwise set it to 0. Bit 2 must be set to 1 if bit 0 (BK Rekeying) is set to 0, i.e., if the packet is not for BK Rekeying, the Authenticator's certificate must be verified. If there are some optional fields in the packet, bit 3 (Optional Field) is set to 1. Otherwise set to 0.
— The Authentication Identifier field is 32 octets in length. If the BK Rekeying is 0, it should be equal to the corresponding field in the Authentication Activation packet sent by Authenticator. If the BK Rekeying is 1, it should be equal to the value negotiated in the last certificate authentication procedure.
— The Supplicant Challenge field is generated randomly by Supplicant, with the length of 32 octets.
— The ECDH Parameter field should be the same as the one in the Authentication Activation packet.
— The Supplicant Key Data field adopts the format defined in 8.1.4.1.1. The content of the field is the temporary public key of the Supplicant used in ECDH exchange.
— The $STA_{Authenticator}$ Identity field adopts the format defined in 8.1.4.1.1. The field contains the ID of $STA_{Authenticator}$.
— The $STA_{Supplicant}$ Certificate field adopts the format defined in 8.1.4.1.1. The field contains the certificate

of $STA_{Supplicant}$.

— The Authenticator Server List trusted by Supplicant field is optional and is encapsulated in the Identity attribute defined in 8.1.4.1.2 (c). The content of the field contains the Authenticator Servers trusted by $STA_{Supplicant}$ except the issuer of the $STA_{Supplicant}$ certificate. If the Supplicant trusts other entities besides its certificate issuer, this field can be used to inform the Authenticator Server about the case.

— The Supplicant Signature field is encapsulated in the Signature attribute defined in 8.1.4.1.2 (a). It is a signature on all the fields excluding the packet header. In computing a signature, the value of the Content subfield in the signature attribute is masked with 0.

After receiving an Authentication Activation from the Authenticator or when a BK rekeying is needed, the Supplicant sends an Access Authentication Request packet to the Authenticator. After the Authenticator receives the Access Authentication Request packet from the Supplicant,

a) If the Authenticator has not sent an Authentication Activation packet, then it checks whether the Authentication Identification field matches the one negotiated in the last Certificate Authentication procedure. If then, the Authenticator proceeds to Step b). Otherwise the Authenticator silently discards the packet. If the Authenticator has sent an Authentication Activation packet and the Authentication Identifier field as well as bit 0 and bit 1 of the FLAG field match those in the Authentication Activation packet sent by the Authenticator, respectively. The Authenticator proceeds to Step b). Otherwise the Authenticator silently discards the packet.

b) In the following cases , the Authenticator silently discards the packet.
   — the Identification field of $STA_{Authenticator}$ does not match its own Identity,
   — the ECDH Parameter field dose not match the one in the Authentication Activation packet,
   — the signature of the Supplicant is not valid
   In the following cases, Authenticator constructs the Certificate Authentication Request packet and sent it to the Authenticator Server.
   — bit 2 in the FLAG field is 1
   — Authenticator requires to verify the $STA_{Supplicant}$ certificate according to its local policy.
   In any other case, the Authenticator proceeds to Step c).

c) If the $STA_{Supplicant}$ certificate is valid, the Authenticator generates a 32-octet random number $N_{Authenticator}$ as a challenge of Authenticator, as well as its own temporary private key y and temporary public key y·P. Then Authenticator obtains the master key seed (x·y·P) abscissa through ECDH computation. By the formula KD_HMAC_SHA256 ((x·y·P) abscissa, NAuthenticator||NSupplicant|| ”base key expansion for key and additional nonce”), the Authenticator computes a 16-octet base key BK and a 32-octet seed. Then, by a SHA256 computation on the seed, a 32-octet Authentication Identifier used in the next Certificate Authentication procedure is obtained. Finally, the Authenticator constructs the Access Authentication Response packet and sent to the Supplicant, in which Access State is set to success and the bit 3 (Optional Field) of FLAG field is set to 0.

   On the other hand, if the $STA_{Supplicant}$ certificate is not valid, the Authenticator constructs the Access Authentication Response packet and sent to the Supplicant, in which Access State is set to failure, the bit 3 (Optional Field) of FLAG field is set to 0, the Challenge $N_{Authenticator}$ field and the Authenticator Key Data (Authenticator temporary public key) field are set to an arbitrary value. Finally, the

Authenticator deauthenticates with $STA_{Supplicant}$.

8.1.4.2.3 Certificate Authentication Request

The Certificate Authentication Request packet is illustrated in Figure 43w and described after the figure.

	ADDID	Authenticator Challenge	Supplicant Challenge	$STA_{Supplicant}$ Certificate	$STA_{Authenticator}$ Certificate	Authenticator Server List trusted by Supplicant
Octets:	12	32	32	variable	variable	variable

Figure43 w—The format of the Data field of Certificate Authentication Request packet

— The ADDID field is assigned by $MAC_{Authenticator}||MAC_{Supplicant}$ with the length of 12 octets.
— The Authenticator Challenge field is a random number generated by Authenticator. It is 32 octets in length.
— The Supplicant Challenge field is 32 octets in length. It should be equal to the one in the Access authentication Request packet.
—The $STA_{Supplicant}$ Certificate field should be equal to the one in the Access Authentication Request packet.
— The $STA_{Authenticator}$ Certificate field adopts the format defined in 8.2.1.4.1.1. The field contains the certificate of $STA_{Authenticator}$.
— The Authenticator Server List trusted by Supplicant field is optional and should be equal to the one in the Access Authentication Request packet.

If the FLAG field in the Access Authentication Request packet indicates a certificate verification request or the Authenticator itself needs a certificate verification, the Authenticator sends a Certificate Authentication Request to the Authenticator Server.

After receiving an Access Authentication Request packet from Supplicant and sending a Certificate Authentication Request packet to the Authenticator Server, the Authenticator will not handle any Access Authentication Request packet sent by the Supplicant until the Certificate Authentication Request packet is timeout.

After the Authenticator Server receives the Certificate Authentication Request packet,

a) Cooperating with other entities, the Authenticator Server verifies the $STA_{Authenticator}$ certificate and the $STA_{Supplicant}$ certificate. If any of the certificates can not be verified, Authenticator Server sets the corresponding verification result to ‘unknown certificate issuer’, and then proceeds to Step b). If both of them can be verified, Authenticator Server verifies the state of $STA_{Authenticator}$ certificate and $STA_{Supplicant}$ certificate, and then proceeds to Step b).
b) According to the result of the $STA_{Authenticator}$ certificate and $STA_{Supplicant}$ certificate verification, the Authenticator Server constructs a Certificate Authentication Response packet with the corresponding signature and sends it to the Authenticator.

8.1.4.2.4 Certificate Authentication Response

The Certificate Authentication Response packet is illustrated in Figure 43

	ADDID	Authentication Result of the Certificate	Server Signature trusted by Supplicant	Server Signature trusted by Authenticator
Octets:	12	variable	variable	variable

Figure 43x—The format of the Data field of Certificate Authentication Response packet

— The ADDID field is assigned by $MAC_{Authenticator}||MAC_{Supplicant}$ with the length of 12 octets. It should be equal to the one in the Certificate Authentication Request packet.

— The Authentication Result of the Certificate is encapsulated in the Certificate Authentication Result attribute defined in 8.2.1.4.1.2 (b). The first nonce in the field is equal to the Authenticator Challenge value in the Certificate Authentication Request packet. The second one is equal to the Supplicant challenge value. The first certificate and its result correspond to the $STA_{Supplicant}$ certificate. The second certificate and its result correspond to the $STA_{Authenticator}$ certificate. The definition of the certificate result is as follows:

0 the certificate is valid
1 the issuer is unknown
2 the certificate is based on an untrusted root
3 the certificate is not time valid
4 the certificate have not a valid signature
5 the certificate is revoked
6 the certificate is not valid for proposed usage
7 the revocation state of the certificate is unknown
8 the certificate has an unknown error

Other values are reserved.

— Server Signature trusted by Supplicant field is encapsulated in signature attribute defined in 8.1.1.4.1.2 (a). It signs the Authentication Result of the Certificate field and itself. When the signature is being calculated, the value of the Content subfield in the signature attribute is masked as 0.

— Server Signature trusted by Authenticator field is encapsulated in signature attribute defined in 8.1.1.4.1.2 (a). It signs all the fields except the ADDID field in this packet, excluding the packet header. When the signature is being calculated, the value of the Content subfield in the signature attribute is masked as 0.

NOTE: If the server trusted by Supplicant is the same as trusted by Authenticator, then the Certificate Authentication Response packet will not include Server Signature trusted by Supplicant field.

When the Authenticator Server receives the Certificate Authentication Request packet, it will send the Certificate Authentication Response packet to the Authenticator.

After the Authenticator receives the Certificate Authentication Response packet,

a) According to the ADDID, the Authenticator identifies the corresponding Certificate Authentication Request packet. If the first nonce in the Authentication Result field in the Certificate Authentication Response packet matches the Authenticator Challenge value in the Certificate Authentication Request packet, proceeds to Step b); otherwise, the Authenticator discards this packet.

b) The Authenticator verifies the signature with the signature list of its trusting Authenticator Server. If the signature is valid, proceeds to Step c). Otherwise, the Authenticator discards this packet.

c) If the $STA_{Supplicant}$ certificate is valid, the Authenticator generates a 32-octet random number $N_{Authenticator}$ as a challenge of Authenticator, as well as its own temporary private key y and temporary public key y·P. Then Authenticator obtains the master key seed (x·y·P) abscissa through ECDH computation. By the formula KD_HMAC_SHA256 ((x·y·P) abscissa, NAuthenticator||NSupplicant|| ”base key expansion for key and additional nonce”), the Authenticator computes out a 16-octet base key BK and a 32-octet seed. Then, by a SHA256 computation on the seed, a 32-octet Authentication Identifier used in the next Certificate Authentication procedure is obtained. Finally, the Authenticator constructs the Access Authentication Response packet sent to the Supplicant, in which Access State is set to success. If the Supplicant requires to verify the STAAuthenticator certificate, the bit 3 (Optional Field) of FLAG field is set to 1 and the Access Authentication Response packet will include Multiple Certificate Verification Result field. Otherwise, the bit 3 (Optional Field) of FLAG field is set to 0 and the Access Authentication Response packet will not include Multiple Certificate Verification Result field.

On the other hand, if the $STA_{Supplicant}$ certificate is not valid, the Authenticator constructs the Access Authentication Response packet and sent to the Supplicant, in which Access State is set to failure, and the Challenge $N_{Authenticator}$ field and the Authenticator Key Data (Authenticator temporary public key) field are set to an arbitrary value. If the Supplicant requires to verify the $STA_{Authenticator}$ certificate, the bit 3 (Optional Field) of FLAG field is set to 1 and the Access Authentication Response packet will include Multiple Certificate Verification Result field. Otherwise, the bit 3 (Optional Field) of FLAG field is set to 0 and the Access Authentication Response packet will not include Multiple Certificate Verification Result field. Finally, the Authenticator deauthenticates with $STA_{Supplicant}$.

8.1.4.2.5 Access Authentication Response

The Access Authentication Response packet is illustrated in Figure 43y and described after the figure.

	FLAG	Supplicant Challenge	Authenticator Challenge	Access Result	Supplicant Key Data	Authenticator Key Data	$STA_{Authenticator}$ Identity	$STA_{Supplicant}$ Identity	Multiple Certificate Verification Result	Authenticator Signature
Octets:	**1**	**32**	**32**	**1**	**Variable**	**Variable**	**Variable**	**Variable**	**Variable**	**Variable**

Figure 43y—The format of the Data filed of Access Authentication Response packet

—The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here only bits 0, 1 and 3 are significant. Bit 0 and bit 1 should have the same value as the FLAG filed in the Access Authentication Request sent by the Supplicant. The Authenticator sets bit 3 (Optional Field) to corresponding value according to the context.

— The Supplicant Challenge field is 32 octets in length. It should be equal to the one in the Access Authentication Request packet.

— The Authenticator Challenge field is 32 octets in length. It should be equal to the one in the Certificate Authentication Request packet.

—The Access Result field is 1 octet in length. The definition is as follows:

0 success, corresponding value of certificate verification result is 0

1 cannot verify the certificate, and corresponding value of certificate verification result is 1

2 certificate error, and corresponding value of certificate verification result is other numerical value than 0 and 1

3 prohibition of the local strategy

Other values are reserved.

— The Supplicant Key Data field adopts the format defined in 8.2.1.4.1.1. The content of the field is the temporary public key of the Supplicant used in ECDH exchange. The field should be equal to the one in the Access Authentication Request packet.

— The Authenticator Key Data field adopts the format defined in 8.2.1.4.1.1. The content of the field is the temporary public key of the Authenticator used in ECDH exchange.

— The $STA_{Authenticator}$ Identity field adopts the format defined in 8.2.1.4.1.1. The field contains the ID of $STA_{Authenticator}$.

— The $STA_{Supplicant}$ Identity field adopts the format defined in 8.2.1.4.1.1. The field contains the ID of $STA_{Supplicant}$.

— The Multiple Certificate Verification Result field is optional. If exists, it contains all the fields except the ADDID with the same content in the Certificate Authentication Response packet.

— The Authenticator Signature field is encapsulated in the Signature attribute defined in 8.2.1.4.1.2 (a). It is a signature on all the fields excluding the packet header. In computing a signature, the value of the Content subfield in the signature attribute is masked as 0.

When the Authenticator receives the Certificate Authentication Responding packet or the Access Authentication request packet, it will send the Access Authentication Response packet to Supplicant.

After the Supplicant receives the Access Authentication Response packet,

a) If the Supplicant conforms that the packet corresponds to the current Access Authentication Requesting packet according to $STA_{Authenticator}$ Identity and $STA_{Supplicant}$ Identity, the Supplicant proceeds to Step b). Otherwise the Supplicant discards the packet.

b) If the bit 0 and bit 1 in the FLAG field match those in the Access Authentication Request packet respectively, the Supplicant proceeds to Step c). Otherwise the Supplicant discards the packet

c) If the Supplicant Challenge field and the Supplicant Key Data match those in the Access Authentication Request packet respectively, the Supplicant proceeds to Step d). Otherwise the Supplicant discards the packet.

d) If the Supplicant does not require the certificate verification in the Access Authentication Requesting packet, the Supplicant proceeds to Step e). Otherwise, the Supplicant will verify the Authenticator Server signature in the Multiple Certificate Authentication Result field. If the verification is not correct, the Supplicant discards the packet. Otherwise, the Supplicant will verify the Authenticator certificate authentication result. If it is not valid, the Supplicant terminates the authentication with the $STA_{Authenticator}$. Otherwise, the Supplicant proceeds to Step e).

e) If the Authenticator signature is not valid, the Supplicant discards the packet. Otherwise, the Supplicant

will verify the Access Result. If it is not "successful," the Supplicant terminates the authentication with the $STA_{Authenticator}$. Otherwise, the Supplicant proceeds to Step f).

f) The Supplicant obtains the master key seed (x·y·P) abscissa through ECDH computation. By the formula KD_HMAC_SHA256 ((x·y·P) abscissa, NAuthenticator||NSupplicant|| "base key expansion for key and additional nonce"), the Supplicant computes out a 16-octet base key BK and a 32-octet seed. Then, by a SHA256 computation on the seed, a 32-octet Authentication Identifier used in the next Certificate Authentication procedure is obtained.

For the ECDH algorithm, there are some explanations as follows:

a) Temporary private keys x and y are the integers between 1 and n-1. n is the degree of the base point p in the elliptic curve domain parameters.

b) Temporary pubic keys p and y.p are the points in the elliptic curve defined in the elliptic curve domain parameters.

c) The key seed $(x \cdot y \cdot P)_{abscissa}$ negotiated by ECDH is the x-coordinate in the x·y·P. x·y·P can not be an infinite point.

8.1.4.3 Unicast key negotiation procedure

Unicast key negotiation procedure uses BK to complete the negotiation of the unicast session key, and constructs the USKSA.

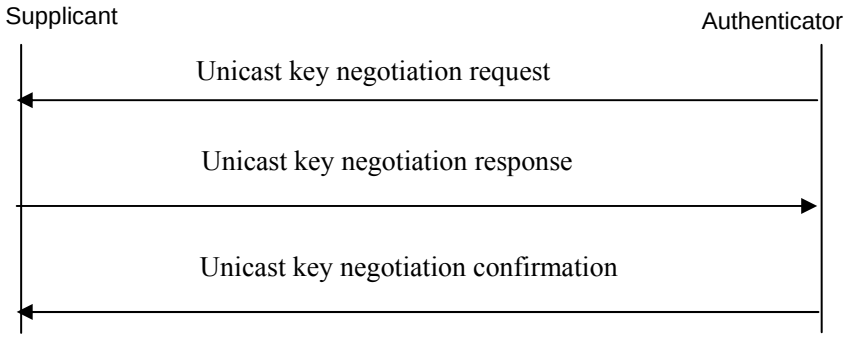

Figure 43z—Unicast key negotiation procedure

8.1.4.3.1 Unicast Key Negotiation Request

The Unicast Key Negotiation Request packet is illustrated in Figure 43aa.

	FLAG	BKID	USKID	ADDID	Authenticator Challenge
Octets:	1	16	1	12	32

Figure 43aa—The format of the Data field of Unicast Key Negotiation Request packet

— The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here, only bit 4 (USK Rekeying) is significant. When the unicast key negotiation procedure is for the initial USKSA establishment, bit 4 (USK Rekeying) is set to 0. When the unicast key negotiation procedure is for USK rekeying, bit 4 (USK

Rekeying) is set to 1

— The BKID field is 16 octets in length, defined in 8.2.1.4.1.1. It indicates the current shared BK.

— The USKID field is 1 octet in length in which the bit 0 is the index of the unicast session key of the negotiation, and all the other bits are reserved. Bit 0 of this field is originally set to 0, and it switches between 0 and 1 after each unicast key negotiation.

— The ADDID field is 12 octets in length, defined in 8.2.1.4.1.1. MAC address 1 in the field is the Authenticator MAC address, and MAC address 2 is the Supplicant MAC address.

— The Authenticator Challenge field is 32 octets in length. If the USK Rekeying is 0, it is a random number generated by Authenticator. If the USK Rekeying is 1, it is equal to the value negotiated in the last unicast key negotiation procedure.

The Authenticator will send a unicast key negotiation request packet to Supplicant and start the unicast key negotiation with Supplicant in the following cases:

— After the Authenticator completes the certificate authentication procedure and constructs the valid BKSA,

— using preshared key authentication method,

— cached BKSA is used,

— rekeying unicast key.

After the Supplicant receives the Unicast Key Negotiation Request packet,

a) If the BKSA referred by BKID is valid and bit 4 of the FLAG field (USK Rekeying) is 1, the Supplicant goes to Step c). If BKSA referred by BKID is valid, bit 4 of the FLAG field (USK Rekeying) is 0 and USKSA referred by USKID is not valid, the Supplicant proceeds to Step b). Otherwise the Supplicant discards the packet.

b) If the Authenticator Challenge is equal to the one negotiated in the last Unicast Key Negotiation procedure, the Supplicant proceeds to Step c). Otherwise the Supplicant discards the packet.

c) The Supplicant generates a random number as the Supplicant challenge. Then the Supplicant generates 96 octets through the formula KD_HMAC_SHA256 (BK, ADDID|| N1||N2||“pairwise key expansion for unicast and additional keys and nonce”) in which N1 is the Authenticator Challenge and N2 is the Supplicant Challenge. The first 64 octets are the unicast session key (the first 16 octets are the unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK). The last 32 octets are the Authenticator Challenge seed of the next unicast session key negotiation procedure. By a SHA256 computation on the seed, a 32-octet Authenticator Challenge used in the next unicast session key negotiation procedure is obtained.

d) The Supplicant calculates the message authentication code by the HMAC_SHA256 algorithm with KCK and constructs the Unicast Key Negotiation Response packet to send to the Authenticator.

e) The Supplicant will install the new unicast session key when it is in a BSS. When the Supplicant is in an IBSS, it shall install the new unicast session key only if its MAC address is less than the Authenticator’s MAC address. The Supplicant should only decapsulate the received unicast data frame with the new unicast session key and must not encapsulate the sent unicast data frame with the new key.

8.1.4.3.2 Unicast Key Negotiation Response

The Unicast Key Negotiation Response packet is illustrated in Figure 43ab.

FLAG	BKID	USKID	ADDID	Supplicant Challenge	Authenticator Challenge	$WIE_{Supplicant}$	Message Authentication Code
Octets: 1	16	1	12	32	32	variable	20

Figure 43ab—The format of the Data field of Unicast Key Negotiation Responding packet

—The FLAG field defined in 8.1.4.1.1 is 1 octet in length. Here, only bits 4 (USK Rekeying) is significant. When the unicast key negotiation procedure is for the initial USKSA establishment, bit 4 (USK Rekeying) is set to 0. When the unicast key negotiation procedure is for USK rekeying, bit 4 (USK Rekeying) is set to 1

— The BKID field is 16 octets in length, defined in 8.1.4.1.1. It indicates the current shared BK .

— The USKID field is 1 octet in length in which the bit 0 is the index of the unicast session key of the negotiation, and all the other bits are reserved. Bit 0 of this field is originally set to 0, and it switches between 0 and 1 after each unicast key negotiation.

— The ADDID field is 12 octets in length, defined in 8.1.4.1.1. MAC address 1 in the field is the Authenticator's MAC address, and MAC address 2 is the Supplicant's MAC address.

— The Authenticator Challenge field is 32 octets in length. If the USK Rekeying is 0, it is equal to the one in the Unicast Key Negotiation Request packet. If the USK Rekeying is 1, it is equal to the one negotiated in the last unicast key negotiation procedure.

— The Supplicant Challenge field is 32 octets in length which is a random number generated by the Supplicant.

— The $WIE_{Supplicant}$ field is the WAPI parameter set chosen by the Supplicant's SME. In the BSS mode, the field is equal to the WAPI parameter set sent by the association request frame from Supplicant. In the IBSS mode, the field contains the unicast cipher algorithm chosen by Supplicant, the multicast cipher algorithm announced by Authenticator, and the currently used authentication and key management suite list.

— The Message Authentication Code field is 20 octets in length. Its value is generated by Supplicant through the HMAC_SHA256 algorithm. The Supplicant calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field. The HMAC algorithm is defined in IETF RFC2104, HMAC: Keyed-Hashing for Message Authentication.

When the Supplicant rekeys, or receives the Unicast Key Negotiation Request, the Supplicant sends the Unicast Key Negotiation Response packet to the Authenticator.

After the Authenticator receives the Unicast Key Negotiation Response packet,

a) If the bit 4 (USK Rekeying) of the FLAG field is 1, the Authenticator proceeds to Step b); otherwise the Authenticator goes to Step c).

b) If there is a valid USKSA and the USKSA referred by USKID in the packet is invalid, the Authenticator proceeds to Step c); otherwise the Authenticator discards the packet.

c) If the Authenticator Challenge is correct, the Authenticator proceeds to Step d). Otherwise the

Authenticator discards the packet.

d) The Authenticator generates 96 octets through the formula KD_HMAC_SHA256 (BK, ADDID|| N1||N2||"pairwise key expansion for unicast and additional keys and nonce") in which N1 is the Authenticator Challenge and N2 is the Supplicant Challenge. The first 64 octets are the unicast session key (the first 16 octets are the unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK). The last 32 octets are the Authenticator Challenge seed of the next unicast session key negotiation procedure. By a SHA256 computation on the seed, a 32-octet Authenticator Challenge used in the next unicast session key negotiation procedure is obtained. The Authenticator calculates the Message Authentication Code locally by the HMAC_SHA256 algorithm. If the calculated value matches that included in the packet, the Authenticator proceeds to Step e); otherwise the Authenticator discards the packet.

e) If the bit 4 (USK Rekeying) of the FLAG field is 1, the Authenticator proceeds to Step f). Otherwise, If the bit 4 (USK Rekeying) of the FLAG field is 0,

— In a BSS, if the $WIE_{Supplicant}$ field is equal to the WAPI parameter set in the received association request frame, the Authenticator proceeds to Step f); otherwise the Authenticator deauthenticates with $STA_{Supplicant}$.

— In an IBSS, if the unicast key algorithm in $WIE_{Supplicant}$ field is supported, the Authenticator proceeds to Step f); otherwise the Authenticator deauthenticates with $STA_{Supplicant}$.

f) The Authenticator calculates Message Authentication Code locally by the HMAC_SHA256 algorithm, and sends the Unicast Key Negotiation Confirmation packet to the Supplicant.

g) The Authenticator will install the new unicast session key when it is in a BSS. When the Authenticator is in an IBSS, it will install the new unicast session key only if its MAC address is greater than the Supplicant's MAC address. The Authenticator should receive and send the unicast data frame with the new unicast session key. For unicast rekeying, the old unicast session key should be deleted if the received unicast data frame is decapsulated correctly with the new key.

8.1.4.3.3 Unicast Key Negotiation Confirmation

The Unicast Key Negotiation Confirmation packet is illustrated in Figure 43ac and described after the figure.

	FLAG	BKID	USKID	ADDID	Supplicant Challenge	$WIE_{Authenticator}$	Message Authentication Code
Octets:	1	16	1	12	32	variable	20

Figure 43ac—The format of the Data field of Unicast Key Negotiation Confirmation packet

— The FLAG field is 1 octet in length, and it is equal to the FLAG field in the Unicast Key Negotiation Response packet.

— The BKID field is 16 octets in length, and it is equal to the BKID field in the Unicast Key Negotiation Response packet.

— The USKID field is 1 octet in length, and it is equal to the USKID field in the Unicast Key Negotiation Response packet.

— The ADDID field is 12 octets in length, and it is equal to the ADDID field in the Unicast Key

Negotiation Response packet.

— The Supplicant Challenge field is 32 octets in length, and it is equal to the Supplicant Challenge field in the Unicast Key Negotiation Response packet.

— The $WIE_{Authenticator}$ field is the WAPI parameter set sent by Authenticator in the Beacon frame and Probe Response frame.

—The Message Authentication Code field is 20 octets in length. Its value is generated by Authenticator through the HMAC_SHA256 algorithm. The Authenticator calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

After the Authenticator receives the Unicast Key Response packet, it will send the Unicast Key Negotiation Confirmation packet to the Supplicant.

After the Supplicant receives the Unicast Key Negotiation Response packet

a) If the Supplicant challenge is equal to the value in the Unicast Key Negotiation Response packet, the Supplicant proceeds to Step b). Otherwise the Supplicant discards the packet.

b) The Supplicant calculates the Message Authentication Code locally by the HMAC_SHA256 algorithm. If the calculated value matches that included in the packet, the Supplicant proceeds to Step c); otherwise the Supplicant discards the packet.

c) If the bit 4 (USK Rekeying) of the FLAG field is 1, the Authenticator proceeds to Step d). Otherwise, if the bit 4 (USK Rekeying) of the FLAG field is 0, and if the $WIE_{Authenticator}$ field is equal to the WAPI parameter set in the received Beacon and Probe Response frame, the Supplicant proceeds to Step d); otherwise the Supplicant deauthenticates with the $STA_{Authenticator}$.

d) The sent unicast data frame will be encapsulated with the new key. For unicast rekeying, the old key should be deleted.

8.1.4.4 Multicast key / STAKey announcement process

The keys negotiated in the unicast key negotiation process are used to achieve the multicast key / STAKey announcement and construct MSKSA or STAKeySA.

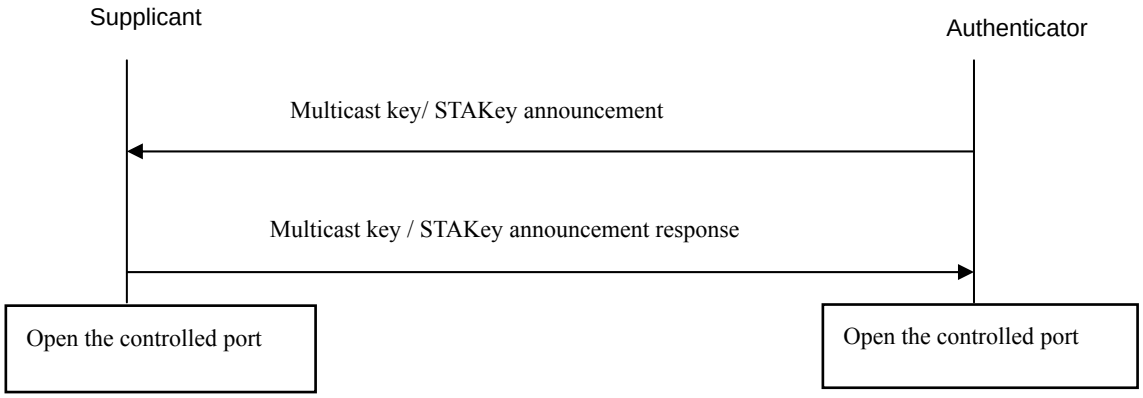

Figure 43ad—Multicast key / STAKey announcement procedure

8.1.4.4.1 Multicast key / STAKey announcement packet

After the success of the unicast key negotiation, or when Authenticator needs to update the multicast key, or

when Authenticator receives the STAKey request packet, Authenticator should send Supplicant the multicast key / STAKey announcement packet to announce the multicast /STAKey master key.

The multicast key / STAKey announcement packet is illustrated in Figure 43ae and described after the figure.

	FLAG	MSKID/STAKeyID	USKID	ADDID	Data packet number	Key Announcement Identifier	Key Data	Message Authentication Code
Octets:	1	1	1	12	16	16	variable	20

Figure 43ae—The format of the Data field of Multicast key / STAKey announcement packet

Here:

— The FLAG field is 1 octet in length, defined as before, and bits 5 and 6 are significant. When the announced key is STAKey, Bit 5 (STAKey negotiation) is set to 1; otherwise it is set to 0. If Bit 6 (STAKey key deletion) is 1, it means to delete the STAKey determined by USKID and ADDID; otherwise it is 0.

— The MSKID/ STAKeyID field is 1 octet in length, in which Bit 0 indicates the currently announced key and all the other bits are reserved. In this field, Bit 0 is initially set to 0, and it will switch between 0 and 1 after each announcement key update.

— The USKID field is 1 octet in length, in which Bit 0 indicates the message authentication key used to calculate the value of the Message Authentication Code field.

— The ADDID field is 12 octets in length. When the announced key is STAKey, the value of this field is: the initiator MAC's address || the peer MAC's address. When announced key is the multicast key, the value of this field is $MAC_{Authenticator}||MAC_{Supplicant}$.

— The Data packet Number field is 16 octets in length, and it indicates the serial number (PN in the WPI data packet) of the data packet that have been encrypted and sent. Hereafter, the serial number of the data frame that STA receives should be larger than the value of this field; otherwise the frame will be discarded.

— The Key Announcement Identifier field is 16 octets in length, and its initial value is 0x5C365C365C365C365C365C365C365C36. During every key updating announcement procedure, the value of this field will increase by 1. If the announced key is unchanged, then the value of this field should keep unchanged. When Authenticator finds that the value of the Key Announcement Identifier field has overflowed, it will deauthenticate with all related STAs. This field is also used as the IV of the key announcement data.

— The Key Data field is defined as before. Its content is the multicast master key or STAKey master key(no MIC) encrypted with the KEK by the negotiated unicast cipher algorithm. The multicast master key or STAKey master key is a 16-octet random number generated by Authenticator, and the IV is the Key Announcement Identifier field.

— The Message Authentication Code field is 20 octets in length. It is calculated by Authenticator through the HMAC_SHA256 algorithm. Authenticator calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

After the Supplicant receives the multicast key / STAKey announcement packet sent by Authenticator it does as follows:

a) If bit 5 of the FLAG field (STAKey negotiation) is 1, the packet is a STAKey announcement packet; otherwise it is a multicast key announcement packet. If Supplicant does not support or not permit STAKey, then when bit 5 of the FLAG field (STAKey negotiation) is 1, this packet is discarded.
   Supplicant uses the message authentication key identified by USKID field to calculate a check value, and compares it with the Message Authentication Code field value. If they are equal, goes to b). Otherwise, this packet is discarded.

b) Checks whether the value of the Key Announcement Identifier field value increases monotonically. If then, goes to c); otherwise, this packet is discarded.

c) Gets a 16-octet announcement master key by decrypting the key data. Then, through KD_HMAC_SHA256 algorithm the Supplicant uses the master key to generate a 32-octet session key (the leading 16 octets are used as the encryption key, and the following 16 octets are used as the integrity check key). If bit 5 of the FLAG field (STAKey negotiation) is 1, then the above key is the STAKey; otherwise, it is the multicast key.

d) Saves the value of the Key Announcement Identifier field, then constructs the multicast key / STAKey response packet and sends it to Authenticator.

e) Installs or deletes the key.
   If this procedure is the multicast key announcement procedure, the new MSK will be installed to receive the multicast data frame. For multicast rekeying, the old MSK should be deleted when a multicast data frame is decapsulated correctly with the new key.
   If this procedure is the STAKey announcement procedure, the initiator/peer will install the new key to send/receive the STA-to-STA data frame. For STAKey rekeying, the old key should also be deleted by the initiator immediately or by the peer when a STA-to-STA data frame is decapsulated correctly with the new key.
   If this procedure is the STAKey deletion procedure, the peer will delete the corresponding STAKey.

If Authenticator announces the new multicast key, Supplicant will save the old multicast key. When it receives the multicast data frame, it will choose the multicast decryption key according to the KeyID field. Once Supplicant receives the multicast data frame from Authenticator with the last announced multicast key, and verifies as well as decrypts it correctly, the Supplicant should discard the old multicast key.

8.1.4.4.2 Multicast key / STAKey response packet

The authentication packet is illustrated in Figure 43af.

	FLAG	MSKID/ STAKeyID	USKID	ADDID	Key Announcement Identifier	Message Authentication Code
Octets:	1	1	1	12	16	20

Figure43af—The format of the Data field of the multicast key/STAKey response packet

Here:

— The FLAG field is 1 octet in length, which is defined as before. This field should be equal to the FLAG field in the multicast key / STAKey announcement packet sent by Authenticator.
— The MSKID/ STAKeyID field is 1 octet, and its value is equal to the one in the multicast key / STAKey announcement packet sent by Authenticator.
— The USKID field is 1 octet in which bit 0 indicates the message authentication key used to calculate the Message Authentication Code field. This field is equal to the USKID field in the multicast key / STAKey announcement packet sent by Authenticator.
— The ADDID field is 12 octets in length, and its value is equal to the one in the multicast key / STAKey announcement packet.
— The Key Announcement Identifier field is 16 octets in length, and its value is equal to the one in the multicast key / STAKey announcement packet sent by Authenticator.
— The Message Authentication Code field is 20 octets in length. Its value is generated by Supplicant through the HMAC_SHA256 algorithm. The Supplicant calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

Supplicant sends Authenticator the multicast key / STAKey response packet.

After the Authenticator receives the multicast key response packet sent by Supplicant, it does as follows:

a) Uses the message authentication key identified by the USKID field to calculate the check value, and then compares it with the value of the Message Authentication Code field. If they are equal, goes to b). Otherwise, discards the packet.
b) Compares the FLAG field, the MSKID/STAKeyID field, the USKID field, the ADDID field and the Key Announcement Identifier field with those in the multicast key/STAKey announcement packet,respectively. If every field is equal to the corresponding field in the announcement packet, then the multicast key/STAKey announcement process is successful. Otherwise, Authenticator discards this packet.
c) Installs the new key when announcing successfully.
   If this procedure is the multicast key announcing procedure, the Authenticator will send multicast data frame encrypted with the new key. When rekeying enabled, the old key should be deleted.
   If this procedure is for the STAKey initiator, the Authenticator will relay the STA-to-STA data frame from initiator to the peer transparently without encrypting and decrypting them.

During the multicast rekeying procedure, Authenticator uses the old multicast key to encrypt the multicast data frame and send it. Authenticator will not use the last announced multicast key until all of the STAs associated with the AP have been successfully announced with the new key.

8.1.4.5 STAKey establishment procedure

When STA1 sends data to STA2 in the same BSS, STA1 could require establishing STAKey with STA2 to encrypt the unicast data from STA1 to STA2. The encryption/decryption algorithm is the multicast cipher suite announced by the AP. The AP will do not encrypt/decrypt the data sent from STA1 to STA2.

This procedure is composed of the multicast key / STAKey announcement procedure and is showed in Figure 43ag.

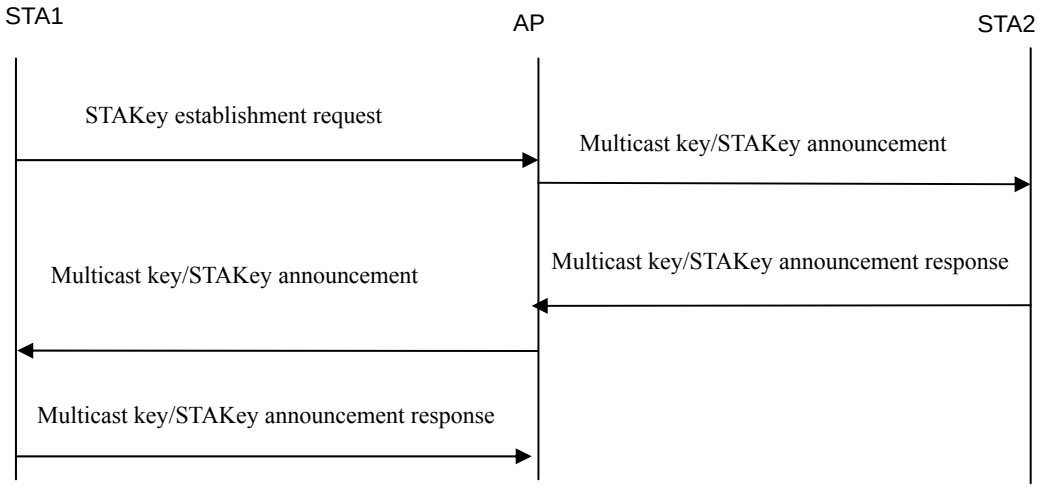

Figure 43ag—The flow chart of the STAKey establishment

STA1 firstly sends the AP the STAKey establishment request. Then the AP uses the random number generation algorithm to generate the STAKey master key and transfers it to STA1 and STA2 through the multicast key / STAKey announcement procedure.

If the AP has announced the STAKey to STA2 successfully, but it is unsuccessful to STA1, then the AP should notify STA2 to delete the corresponding STAKey through the multicast key / STAKey announcement procedure, and the AP should set the STAKey deletion flag to 1.The deletion flag is in the FLAG field of the multicast key / STAKey announcement packet.

If STA1 delete some STAKey, it should set the STAKey deletion flag in the FLAG field of the STAKey establishment request packet to 1.

The STAKey establishment request packet is illustrated in Figure 43ah.

	FLAG	STAKeyID	USKID	ADDID	Replay Counter	Message Authentication Code
Octets:	1	1	1	12	16	20

Figure 43ah—The format of the Data field in the STAKey establishment request packet

Here:

— The FLAG field is 1 octet in length, and is defined as before. Here, bits 5 and 6 are significant, and bit 5 (STAKey negotiation) is set to 1. When establishing the STAKey, bit 6(STAKey deletion) is set to 0; and when deleting the STAKey, bit 6 is set to 1.

— The STAKeyID field is 1 octet in length, and is set to the key index value chosen by the initiator.

— The USKID field is 1 octet in length, in which bit 0 indicates the message authentication key used to calculate the Message Authentication Code field. The value of the USKID field should be equal to the

one in the multicast key announcement packet sent by the AP.

— The ADDID field is 12 octets in length, and its value is the initiator's MAC address || the peer's MAC address.

— The Replay Counter field is 16 octets in length, with the initial value 1. During each rekeying announcement, its value increases by 1.

— The Message Authentication Code field is 20 octets in length. Its value is generated by STA through the HMAC_SHA256 algorithm. STA calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

8.1.4.6 Pre-authentication

If the destination AP supports the pre-authentication, the STA can execute the pre-authentication with the destination AP through the currently associated AP. The STA requires to start the pre-authentication by sending a pre-authentication start packet to the destination AP. When the currently associated AP receives the pre-authentication start packet, if the Replay Counter and the Message Authentication Code are valid, then it forwards the packet to the destination AP. When the destination AP receives this packet, it starts the WAI certificate authentication procedure by sending the authentication activation packet to the corresponding STA identified by the ADDID field.

The pre-authentication start packet is illustrated in Figure 43ai .

	FLAG	USKID	ADDID	Replay Counter	Message Authentication Code
Octets:	1	1	2	16	20

Figure 43ai—The format of the Data field in the pre-authentication start packet

Here:

— The FLAG field is 1 octet in length, and is defined as before.

— The USKID field is 1 octet in length, where bit 0 indicates the message authentication key used in calculating the Message Authentication Code field. The value of this field is equal to the one in the USKID field of the multicast key announcement packet sent by the AP.

— The ADDID field is 12 octets in length, and its value is the initiator STA's MAC address || the destination AP's MAC address.

— The Replay Counter field is 16 octets in length, with the initial value 1. Each time the STA sends the pre-authentication start packet the value of this field increases by 1. This Replay Counter field is same as that in the STAKey establishment request packet.

— The Message Authentication Code field is 20 octets in length. Its value is generated by the STA through the HMAC_SHA256 algorithm. The STA calculates the value over the content of all protocol data fields before this field (excluding the packet header) with the message authentication key identified by the USKID field.

The STA start the pre-authentication procedure with other APs only after the STA finishes the negotiation and the unicast session key installation with the currently associated AP. If the destination AP declares to support the

pre-authentication in the WAPI Capability Information field of the WAPI Information Element, the STA can start the pre-authentication procedure with this AP. Only if the certificate authentication and key management method is adopted, the pre-authentication procedure can be started.

The Supplicant of STA activates the authentication procedure by sending a WAI “pre-authentication start” packet to the associated AP. In this packet, the DA is set to the destination AP’s BSSID and the RA is set to the currently associated AP’s BSSID. The destination AP should use the BSSID as Authenticator’s MAC address. The currently associated AP should forward the pre-authentication packet sent by the STA to DS, and send the pre-authentication packet sent by the DS to STA.

The destination AP that receives the “pre-authentication start” packet through DS should start the authentication procedure with the corresponding STA. DS should forward the authentication message to the associated AP.

If the WAI certificate authentication procedure succeeds, the result of the pre-authentication is BKSA. If the STA has been associated to the pre-authenticated AP, the STA can use the BKSA to start the unicast key negotiation procedure and multicast key announcement procedure. If the cached status of the pre-authentication result between the AP and STA is not synchronized, the unicast key negotiation procedure will fail. In this situation, the value of dot11WAPIStatsWAIUnicastHandshakeFailures MIB will increase by 1.

Whether the destination AP is in the signal range of STA or not, if only the AP supports pre-authentication, the Supplicant of the STA can execute pre-authentication procedure with the AP.

The AP can possible delete the pre-authentication result BKSA because of the limitation of the resources or timeout, therefore even if the pre-authentication procedure has already succeeded, the STA may require executing the WAI certificate authentication and the key negotiation procedure after associated to the destination AP.

NOTE: All WAI packets (including all packets in the certificate authentication, the unicast key negotiation, the multicast key announcement and the STAKey establishment procedures) are transferred by 0x88B4 protocol. If the format of a packet is incorrect, the MIB value dot11WAPIStatsWAIFormatErrors should be increased by 1; If the result of the signature verification is incorrect, the MIB value dot11WAPIStatsWAISignatureErrors should be increased by 1; If the result of the message authentication code check is incorrect, the MIB value dot11WAPIStatsWAIHMACErrors should be increased by 1.

8.1.4.7 Cached BKSAs and WAPI key management

A STA can cache BKSAs between a STA and AP. The BKSAs cannot be changed when being cached.

If a STA in an ESS has a valid BKSA with an AP to be (re) associated, it includes the BKID of the BKSA in the WAPI Information Element of the (Re) Association Request. After receives a (Re) Association Request with one or more BKIDs, an AP checks whether it has cached a BKSA for the BKID and whether the BKSA is still valid. If so, it starts the unicast session key negotiation procedure and the multicast key announcement procedure; otherwise it starts a full authentication and key negotiation procedure after association.

If a STA roams to an AP which has not completed the pre-authentication, and the STA does not have a BKSA for that AP, the STA should start a full WAI certificate authentication and key negotiation procedure.

8.1.4.8 Rekeying

An AP can start the updating of the base key, the unicast key and the multicast key, and the STA can start the updating of the base key and the unicast key. The identified key index in rekeying process must be the currently invalid key index, and the old key must be deleted after the new key is activated.

Once the base key is updated, the unicast key must be updated as well. The process from the base key update beginning to the unicast key update completing must be completed within the time set by the MIB value dot11WAPIConfigSATimeout, otherwise the two STAs will be deauthenticated.

During the unicast rekeying procedure, there is no confirmation of the WAPI parameter set or selection of the unicast ciper algorithm, the STA still use the unicast cipher algorithm negotiated in the initial unicast cipher negotiation procedure.

8.1.4.9 Timeout processing

After the STA and AP send the message, if they cannot receive any response within a timeout, then increase the MIB value dot11WAPIStatsWAITimeoutCounters by 1.
If the message is a certificate authentication packet, then re-send the message "dot11WAPIConfigCertificateUpdateCount" times.
If the message is a unicast key negotiation packet, then re-send the message "dot11WAPIConfigUnicastUpdateCount" times.
If the message is a multicast key / STAKey announcement packet, then re-send the message "dot11WAPIConfigMulticastUpdateCount" times.
The default timeout value of the certificate authentication request packet is set to 10 seconds, that of the access authentication request packet is 31 seconds, and others are 1 second.

During the certificate authentication procedure, the unicast key negotiation procedure and multicast key announcement procedure, if there is still no response after re-sending the message certain times, then the STA and AP will terminate the authentication.
If the procedure is a certificate authentication procedure, then increases the MIB value dot11WAPIStatsWAICertificateHandshakeFailures by 1.
If the procedure is a unicast key negotiation procedure, then increases MIB value dot11WAPIStatsWAIUnicastHandshakeFailures by 1.
If the procedure is a multicast key announcement procedure, then increases the MIB value dot11WAPIStatsWAIMulticastHandshakeFailures by 1.

During the STAKey announcement procedure, if there is still no response after re-sending the message certain times, then increases the MIB value dot11WAPIStatsWAIMulticastHandshakeFailures by 1.

8.1.4.10 Key derivation architecture

In the WAI authentication and key management systems, the KD-HMAC-SHA256 is used to derive every key and challenge from the corresponding master key.

8.1.4.10.1 BK key derivation architecture

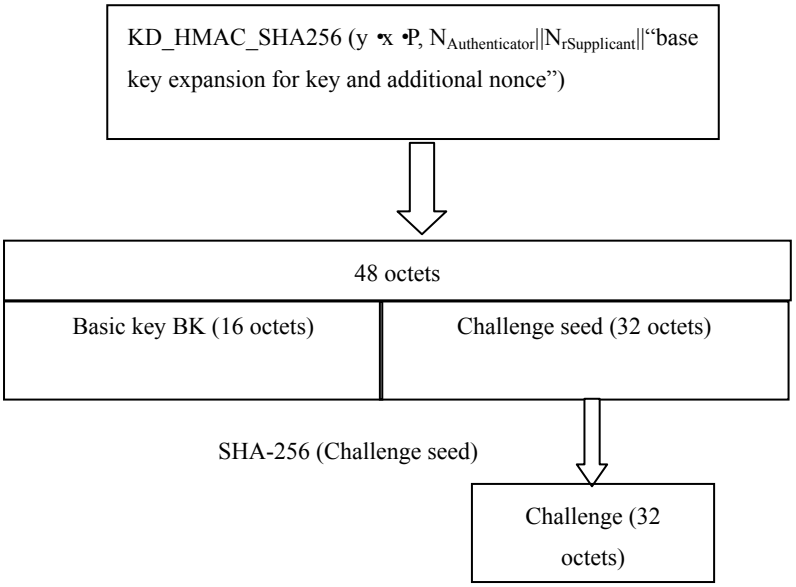

Figure 43aj—BK key derivation architecture

8.1.4.10.2 Unicast key derivation architecture

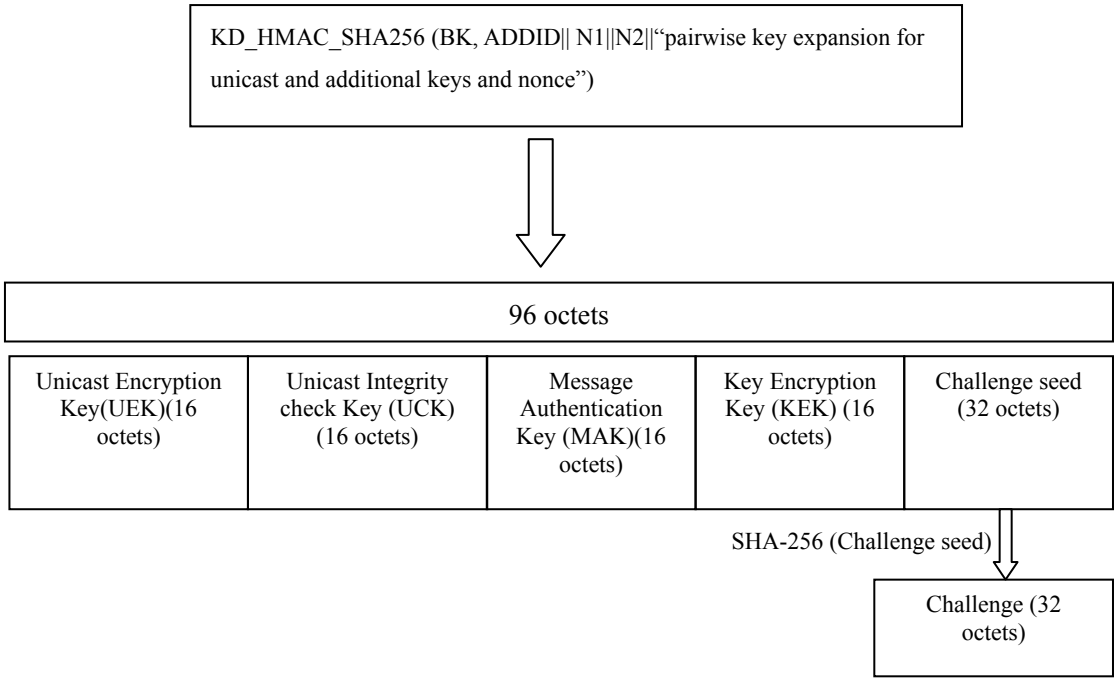

Figure 43ak—Unicast key derivation architecture

8.1.4.10.3 Multicast / STAKey derivation architecture

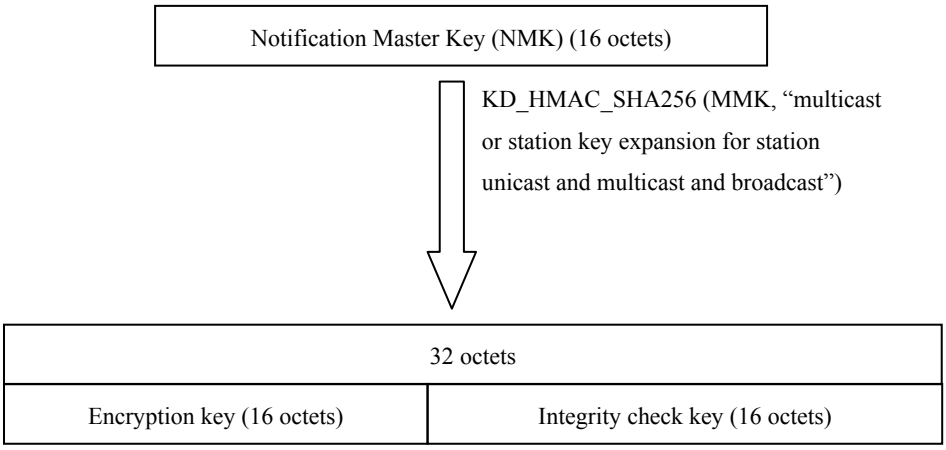

Figure 43al—Multicast / STAKey derivation architecture

8.1.4.10.4 Preshared key derivation architecture

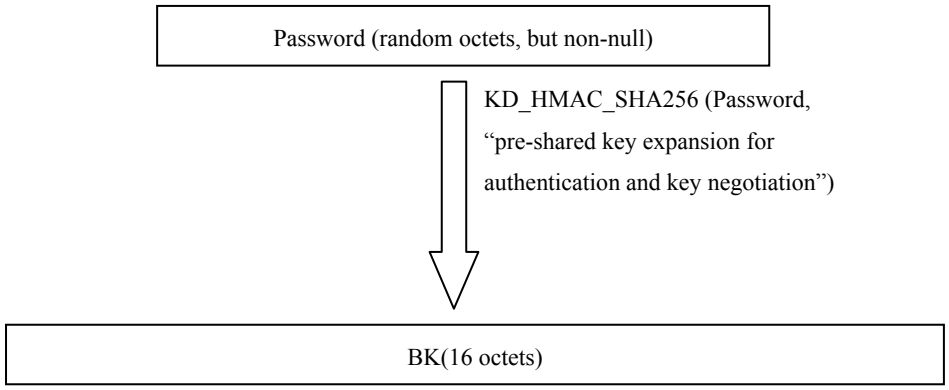

Figure 43am—Preshared key derivation architecture

8.1.4.11 WAI protocol packet’s fragmentation and defragmentation

When the STA or AP has a WAI protocol packet to send, it needs to determine which local port to send the packet, and then query this port to get its MTU (Maximum Transmission Unit). Comparing the packet length with the MTU, if the packet length is larger than the MTU, the sender has to fragmentate the packet. Each fragment has its own WAI protocol header. All fragments except the last one should have the same length, and their length should be a maximum value smaller than the MTU. Here we suggest this value is multiple of 8 octets.

When these fragments reach the destination receiver, it is necessary to reassemble them. The information contained in the WAI protocol fragment header is enough for the receiver to correctly reassemble these fragments into a packet.

The WAI protocol layer uses three fields in the protocol packet header to fragmentate and defragmentate.
For each WAI protocol packet from a sender, its WAI packet serial number field includes a unique value (the

serial number of the first WAI protocol packet is 1, afterwards the serial number of the WAI protocol packet increases by 1), that value is copied to every fragment during the WAI protocol packet fragmentation. If a packet is a retransmitted one, its packet serial number keeps unchanged.
The fragment serial number field denotes the sequence number of the fragment (the serial number of the first fragment is 0, afterwards the serial number of the fragment increases by 1).
One bit in the fragment flag field is used to indicate “more fragments”. In all fragments except the last one of the protocol packet, this bit should be set to 1; the last fragment of the protocol packet, this bit should be set to 0.
Moreover, when a protocol packet is fragmented, the length field of each fragment denotes the length of the fragment, i.e. the total length of the WAI protocol header and the fragment data (in octet).

Fragmentation and defragmentation should be completed in the WAI protocol layer, and the MAC layer should operate in the strict order mode in these processes. Whether the sender is the STA or AP, it should send the packets according to the packet serial number from small to large in the protocol layer. When the fragmentation appears, the fragments of the same WAI protocol packet should be sent according to fragment serial number from small to large. During the receiving, if the receiver protocol layer has received all fragments of the same WAI protocol packet, it will reassemble them to recover the original protocol packet and process. If the received WAI protocol fragments have wrong sequence, discard all the fragments of the currently received WAI protocol packet. If the received WAI protocol packet has wrong sequence, discard the last former WAI protocol packet.

The timeout re-transmission mechanism of each protocol packet in the WAI protocol (the certificate authentication and key negotiation)has been already defined, therefore it is not necessary to define each fragment timeout re-transmission mechanism, i.e., in a WAI protocol packet transmission procedure, even if there is only one fragment lost, the whole protocol packet should be re-transmitted.

8.1.4.12 Port control and data transmission

The WAI controlled port status is *off* at the beginning. When the unicast key negotiation is finished, the STA uses the MLME-SETWPIKEYS.request primitive to set the unicast key. When the multicast key announcement is finished, the STA uses the MLME-SETWPIKEYS.request primitive to set the multicast key. At the same time, the STA uses the MLME-SETPROTECTION.request primitive to start the WPI encrypting /decrypting function for unicast data and multicast data, and the STA sets the WAI controlled port status to *on*.

During the updating of BKSA, USKSA and MSKSA, the WAI controlled port status is kept on.

Whether WPI is enabled or not, the WAI protocol packet will be transmitted in the plaintext mode.

8.2 WPI privacy infrastructure

WPI encrypts MPDUs (except the MPDUs of the WAI protocol packet) with the block cipher of 128-bit size and 128-bit key in the MAC sub-layer. Operation mode and encapsulation structure of the block cipher are illustrated as follow.

8.2.1 Opertate mode

In WPI, the integrity check algorithm works in the CBC-MAC mode, and the data confidentiality uses the

symmetric encryption algorithm operating in the OFB mode. The CBC-MAC and OFB mode is defined in ISO/IEC 10116: 1997 (2nd edition) Information technology - Security techniques - Modes of operation for an n-bit block cipher algorithm.

These two modes are illustrated as follow:

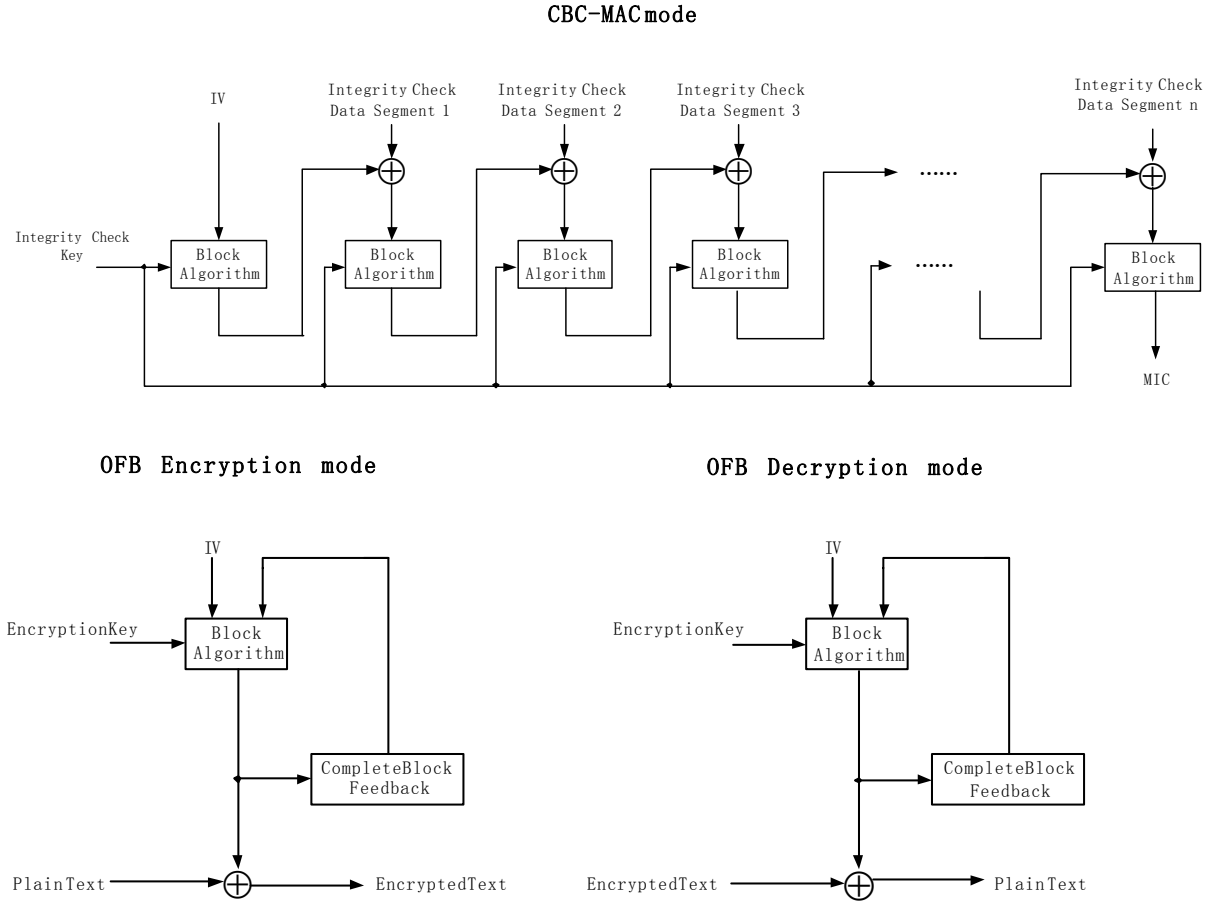

Figure 43an—Operate modes

8.2.2 Key

Key management based on a certificate

During the certificate authentication procedure, Supplicant and Authenticator firstly use the ECDH exchange and the KD_HMAC_SHA256 algorithm to negotiate a 16-octet BK. Then during the unicast key negotiation procedure, Supplicant and Authenticator respectively exchange a random number and use the KD_HMAC_SHA256 algorithm and the BK to generate 96 octets. The first 64 octets are the unicast session key (the first 16 octets are unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK), the last 32 octets are the challenge seed for the next unicast session key negotiation procedure. Then use SHA-256 function, with the seed to obtain a 32-octet challenge for the next unicast key negotiation procedure, and save it. At last, during the multicast key/STAKey announcement procedure, Supplicant and Authenticator will respectively use the KD_HMAC_SHA256 algorithm to extend the 16-octet announcement master key to generate a 32-octet session key (the first 16 octets

are the encryption key, and the last 16 octets are the integrity check key).

Key management based on the preshared key

In the preshared key mode, Supplicant and Authenticator will use the shared key as the seed to generate the BK. During the unicast key negotiation procedure, Supplicant and Authenticator respectively exchange a random number and use the KD_HMAC_SHA256 algorithm and the BK to generate 96 octets. The first 64 octets are the unicast session key (the first 16 octets are unicast encryption key, the second 16 octets are the unicast integrity check key, the third 16 octets are KCK, and the fourth 16 octets are KEK), the last 32 octets are the challenge seed for the next unicast session key negotiation procedure. Then use the SHA-256 function, with the seed to obtain a 32-octet challenge for the next unicast key negotiation procedure, and save it. At last, during the multicast key/STAKey announcement procedure, Supplicant and Authenticator will respectively use the KD_HMAC_SHA256 algorithm to extend the 16-octet announcement master key to generate a 32-octet session key (the first 16 octets are the encryption key, and the last 16 octets are the integrity check key).

NOTE: In a BSS, the key management based on the certificate or the key management based on the preshared key can be adopted. In the IBSS mode, the key management based on the certificate or the key management based on the preshared key can be used, and the initiator STA and the peer STA are respectively acted as Authenticator and Supplicant, and each of them is required to be the initiator to complete a unicast key negotiation procedure and a multicast key announcement procedure.The unicast data key negotiated in the process initiated by the STA with a larger MAC address in chose as the data transmission key. In the WDS mode, only the key management based on the preshared key can be adopted, and its procedure is the same as that in IBBS mode. Currently, the WDS privacy communication supporting is optional

The MAC address is 6 octets. For the purposes of comparison, the MAC address is encoded as 6 octets, taken to represent an unsigned binary number. The first octet of the MAC address shall be used as the most significant octet.

When the user inputs the shared master key (password), it needs to support both the Hex mode and the ASCII Character mode.

8.2.3 Encapsulation and decapsulation

The encapsulation structure of WPI is as follow.

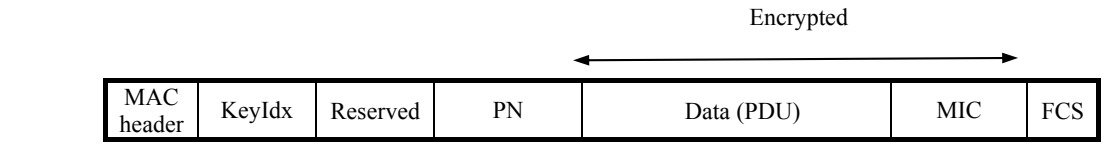

Figure 43ao—WPI's MPDU encapsulation structure

Here,

— The MAC Header field is 30 octets in length if the address **A4** exists; otherwise it is 24 octets in length.

— The KeyIdx field is one octet in length, and indicates the USKID, MSKID, or STAKeyID.

— The Reserved field is one octet in length, with the default value 0.

—The PN field is 16 octets in length, and it indicates the serial number of the data packet.

—The PDU (data) field indicates the MPDU data, and its maximum length is 2278(2312-18-16).

—The MIC field is 16 octets in length, and its value is calculated over the integrity check data using the integrity check key in the CBC-MAC mode, the IV required for the calculation is the value of the PN field in the packet.

—The FCS field is 4 octets in length.

The integrity check data is defined as below:

	FC	A1	A2	SC	A3	A4	KeyIdx	Reserved	L	Data (PDU)
Octets:	2	6	6	2	6	6	1	1	2	>0

Figure 43ap—Integrity check data

— The MAC Header field: the Frame Control field **FC** (Bits 4, 5, 6, 11, 12 and 13 are 0; bit 14 is1) with a length of 2 octets; address **A1** with the length of 6 octets; address **A2** with the length of 6 octets; address **A3** with the length of 6 octets; the Sequence Control field **SC** (bits 4—15 are 0) with the length of 2 octets; address **A4** with the length of 6 octets. Altogether, they are 28 octets in length.

NOTE: if address **A4** does not exist in the MAC header, the corresponding 6 octets are set to 0.

— The KeyIdx field, with the length of one octet.

— The Reserved field, with the length of one octet, with the default value 0.

— The length **L** of the MPDU data, with the length of 2 octets.

— The MPDU data, with the length of more than 0 octet.

In the WPI, when using CBC-MAC to calculate MIC, if the lenth of the integrity check data is not multiple of 16 octets, then expend the integrity check data to the multiple of 16 octets by adding as less as possible at the end. The receiver will use the same method to verify.

During the data transmission, the WPI's MPDU encapsulation procedure is as follow, where "||" is the concatenation operation:

a) Using the integrity check key and the data packet serial number PN, calculate over the integrity check data to get the integrity check code MIC in CBC-MAC mode;

b) Using the encryption key and the data packet serial number PN, encrypt MPDU data || MIC to obtain the ciphertext in OFB mode;

c) Encapsulate and send it.

During the data reception, the WPI's MPDU decapsulation procedure is as follow:

a) Verify whether PN is valid. If it is invalid, then discard the data, and increase the MIB value dot11WAPIStatsWPIReplayCounters by 1;

b) Using the decryption key and PN, decrypt MPDU data || MIC ciphertext to recover the plaintext in OFB mode. If there is no valid decryption key, then discard the data and increase the MIB value dot11WAPIStatsWPIDecryptableErrors by 1;

c) Using the integrity check key and PN, locally calculate over the integrity check data to obtain the integrity check code MIC in CBC-MAC mode. If the calculated MIC is not equal to the received MIC,

then discard the data and increase the MIB value dot11WAPIStatsWPIMICErrors by 1;

d) Decapsulate and reassemble the MPDU plaintext.

8.2.4 Rules for using data packet serial number PN

In the unicast session

After each unicast key update,

Supplicant initializes the value of PN as 0x5C365C365C365C 365C365C365C365C36. Before Supplicant sends each unicast data frame, it should firstly increase the value of PN by 2.

Authenticator initializes the value of PN as 0x5C365C365C365C365C365C365C365C37. Before Authenticator sends each unicast data frame, it should firstly increase the value of PN by 2.

When Supplicant receives the unicast data frame from Authenticator, it should verify whether the value of PN corresponding to the USKID increases strictly monotonically and whether it is an odd number. If not, discard the packet.

When Authenticator receives the unicast data frame from Supplicant, it should verify whether the value of PN corresponding to the USKID increases strictly monotonically and whether it is an even number. If not, discard the packet.

Authenticator can update the unicast key by the policy based on time or data packet amount. Besides, the overflow problem of PN value needs to be solved by the unicast key update initiated by Authenticator.

In the multicast session

After each multicast key update, Authenticator initializes the value of PN as 0x5C365C365C365C365C365C365C365C36. Before Authenticator sends each multicast data frame, it should firstly increase the value of PN by 1.

When Supplicant receives the multicast data frame, it should verify whether the value of PN corresponding to the MSKID increases strictly monotonically. If not, discard the packet.

Authenticator can update the unicast key by the policy based on time or data packet amount. Besides, the overflow problem of PN value needs to be solved by the unicast key update initiated by Authenticator.

In the STA-to-STA session

After each STA-to-STA key establishment, the initiator of the STAKey negotiation initializes the value of PN as 0x5C365C365C365C365C365C365C 365C36. Before the initiator sends each unicast data frame to the peer part, it should firstly increase the PN value by 1.

When the peer STA receives the unicast data encrypted with the STAKey, it should verify whether the value of PN corresponding to STAKeyID increases strictly monotonically. If not, discard that packet.

The initiator of STAKey can update the STAKey by the policy based on time or the data packet amount. Besides,

the overflow problem of the PN value needs to be solved by the STA-to-STA key update.

8.3 WAPI Authentication and key management state machine

8.3.1 WAPI Supplicant Authentication and key management state machine

The Supplicant shall reinitialize the Supplicant state machine whenever its system is initialized. A Supplicant enters the AUTHENTICATION state on an event from the MAC that requests another STA to be authenticated. If the MIC or any of the WAI packets fails, the Supplicant silently discards the packet. Figure 43aq and Figure 43ar depict the Supplicant state machine.

The management entity will send an authentication request event when it requires an Authenticator authenticated. This can be before or after the STA associates to the AP. In an IBSS environment, the event will be generated when a Probe Response frame is received.

Unconditional transfer (UCT) means the event triggers an immediate transition.

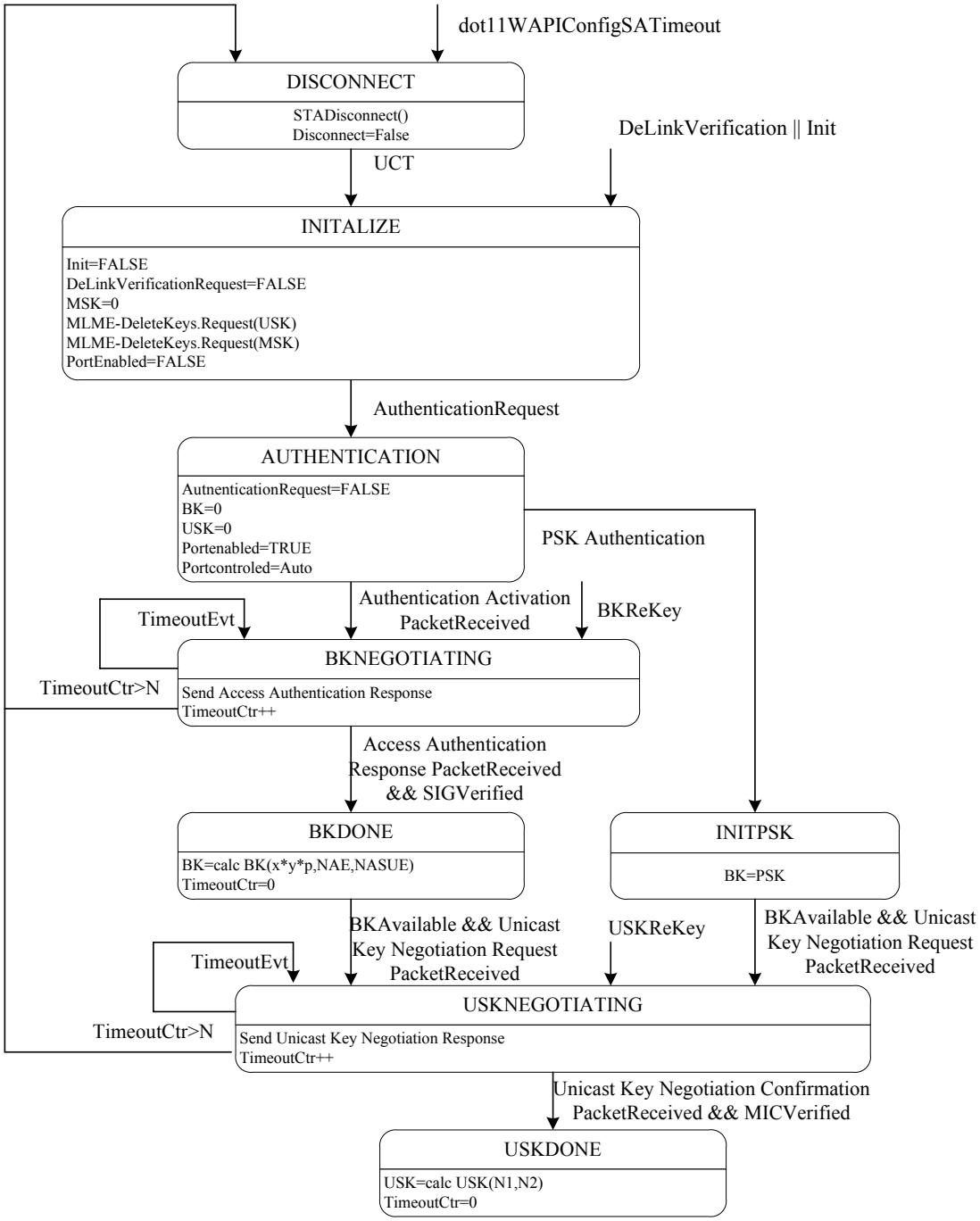

Figure 43aq—Supplicant state machine, Part 1

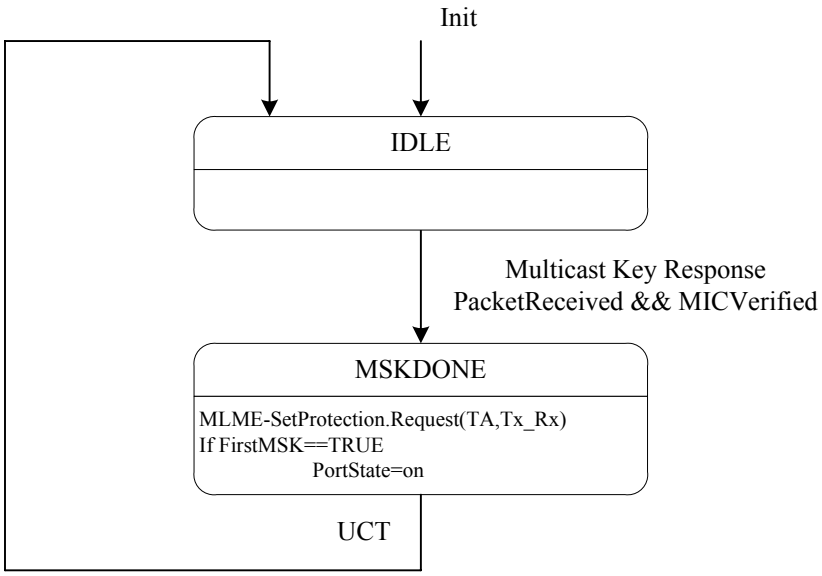

Figure 43ar—Supplicant state machine, Part 2

8.3.1.1 Supplicant state machine states

The following list summarizes the states of the Supplicant state machine:

— **DISCONNECT:** A STA's Supplicant entered this state when WAI procedure is timeout. The Supplicant executes StaDisconnect and enters the INITIALIZE state.

— **INITIALIZE:** A STA's Supplicant enters this state from the DISCONNECT state, when it receives Disassociation or Deauthentication messages or when the STA initializes, causing the STA's Supplicant to initialize the key state variables.

— **AUTHENTICATION**: This state is entered when an AuthenticationRequest is sent from the management entity to authenticate a BSSID.

— **INITPSK:** This state is entered when a PSK is configured.

— **BKNEGOTIATING:** This state is entered when the Authentication Activation packet is received or if BKReKey is set TRUE.

— **BKDONE:** This state is entered when the Access Authentication Response packet is received and signature is verified.

— **USKNEGOTIATING:** This state is entered from BKDONE to start the USK Negotiation when BKAvailable is set to TRUE and Unicast Key Negotiation Request packet is received or if USKReKey is set to TRUE.

— **USKDONE:** This state is entered when the Unicast Key Negotiation Confirmation packet is received and MIC is verified.

— **MSKDONE:** This state is entered when the Multicast Key Announcement packet is received from the Authenticator and MIC is verified.

— **IDLE:** This state is entered when no Multicast Key Announcement is occurring.

8.3.1.2 Supplicant state machine variables

The following list summarizes the variables used by the Supplicant state machine:

— *AuthenticationRequest* – This variable is set to TRUE if the STA's ISO/IEC 8802.11 management entity requires an association to be authenticated. This can be set when the STA associates or at other time.

— *DeauthenticationRequest* – This variable is set to TRUE if a Disassociation or Deauthentication message is received.

— *Disconnect* – This variable is set to TRUE when the STA should initiate a Deauthentication.

— *PacketReceived* – This variable is set to TRUE when a WAI packet is received.

— *Init* – This variable is used to initialize per-STA state machine.

— *TimeoutEvt* – This variable is set to TRUE if the WAI packet sent out fails to obtain a response from the Authenticator. The variable may be set by management action or set by the operation of a timeout while in the USKNEGOTIATING and BKNEGOTIATING states.

— *TimeoutCtr* – This variable maintains the count of WAI packet receive timeouts. It is incremented each time a timeout occurs on a WAI packet receive event and is initialized to 0.

— *HMACVerified* – This variable is set to TRUE if the MIC on the received WAI packet is verified correct. Any WAI packets with an invalid HMAC will be dropped and ignored.

— *SIGVerified* – This variable is set to TRUE if the signature on the received WAI packet is verified correct. Any WAI packets with an invalid signature will be dropped and ignored.

— *USK* – This variable is the current USK.

— *MSK* – This variable is the current MSK.

— *BK* – This variable is the buffer holding the current BK.

— *BKReKey* – This variable is set to TRUE when a BK rekeying is required.

— *USKReKey* – This variable is set to TRUE when a USK rekeying is required.

—*Certificate_Verificaton*– This variable is set to TRUE when the certificate is required to be verified.

—PortControlled – This variable is the method by which the port is controlled. The value is Auto or ForceUnauthenticated.

—PortEnabled – This variable is set to TRUE if the port control mechanism is applied.

—PortState – This variable is set to on if the data frame is permitted to pass through the port.

—Certificate Authentication– This variable is set to TRUE if the certificate authentication and key management is chosen.

—PSK Authentication– This variable is set to TRUE if the PSK authentication and key management is chosen.

8.3.1.3 Supplicant state machine procedures

The following list summarizes the procedures used by the Supplicant state machine:

— **STADisconnect**() – Execution of this procedure Deauthenticates the STA.

8.3.2 WAPI Authenticator Authentication and key management state machine

There is one state diagram for the Authenticator. In an ESS, the Authenticator will always be on the AP; and in an IBSS environment, the Authenticator will be on every STA.

The state diagram shown in parts in Figure 43as through Figure 43awconsists of the following states:

a) The AUTHENTICATION, INITPSK, BKSTART, BKNEGOTIATING, BKNEGOTIATING2, BKNEGOTIATING3, BKDONE, USKSTART, USKNEGOTIATING, USKDONE, DISCONNECT, DISCONNECTED, and INITIALIZE states. These states handle the initialization, certificate authentication, unicast key negotiation, tear-down, and general clean-up. These states are per associated STA.

b) The IDLE, MSKNEGOTIATING, KEYERROR, MSKDONE states. These states handle the transfer of the MSK to the associated client. These states are per associated STA.

c) The MSK_INIT, SETKEYS, and SETKEYSDONE states. These states change the MSK when required, trigger all the USK multicast key state machines, and update the ISO/IEC 8802.11 MAC in the Authenticator's AP when all STAs have the updated MSK. These states are global to the Authenticator.

When the MSK is to be updated the variable MSKReKey is set. The SETKEYS state updates the MSK and triggers all the USK multicast key state machines that currently exist—one per associated STA. Each USK multicast key state machine sends the MSK to its STA. When all the STAs have received the MSK (or failed to receive the key), the SETKEYSDONE state is executed which updates the APs encryption/integrity engine with the new key.

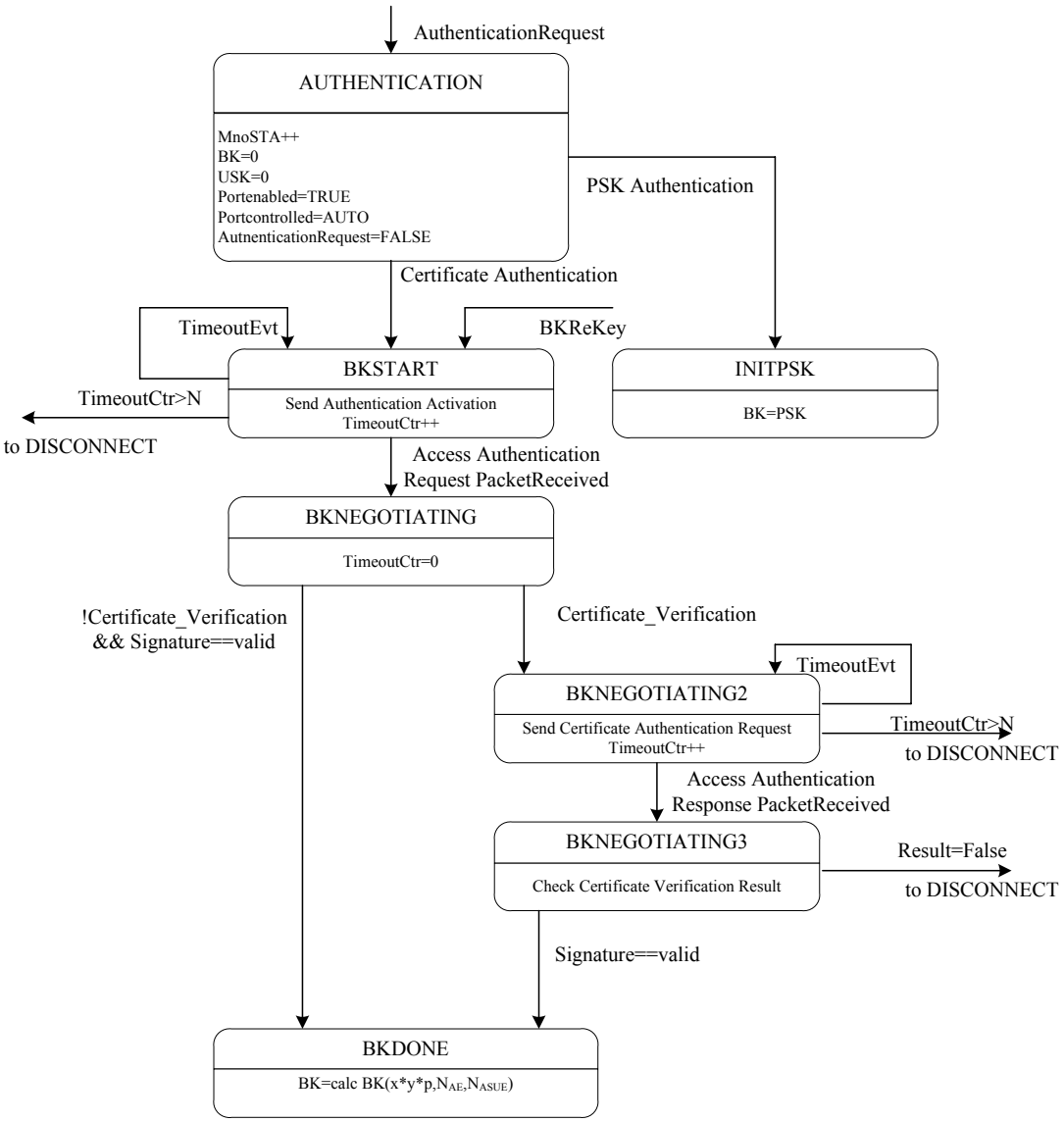

Figure 43as—Authenticator state machine, Part 1

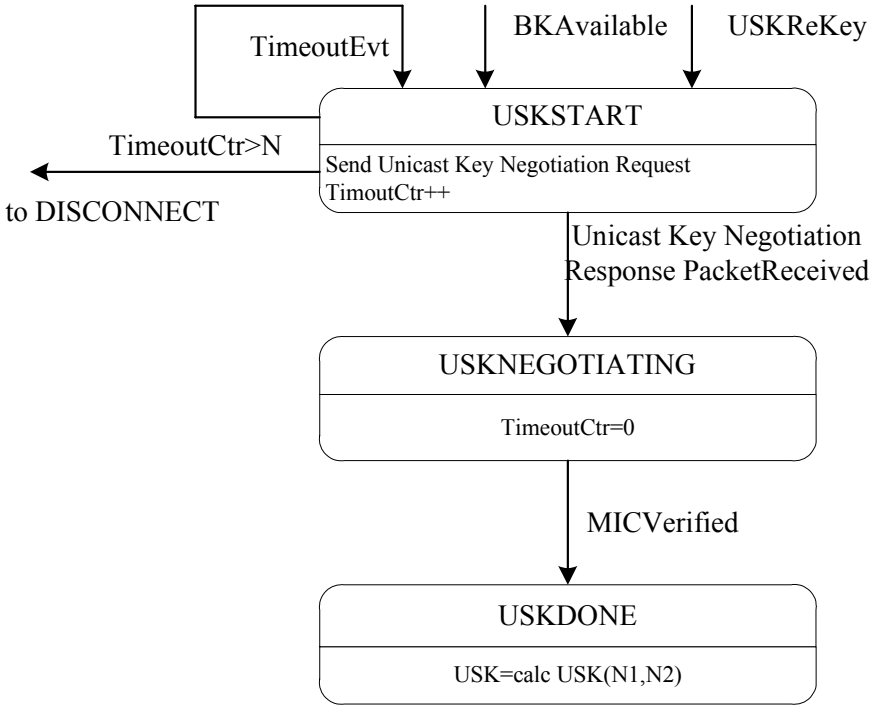

Figure 43at—Authenticator state machine, Part 2

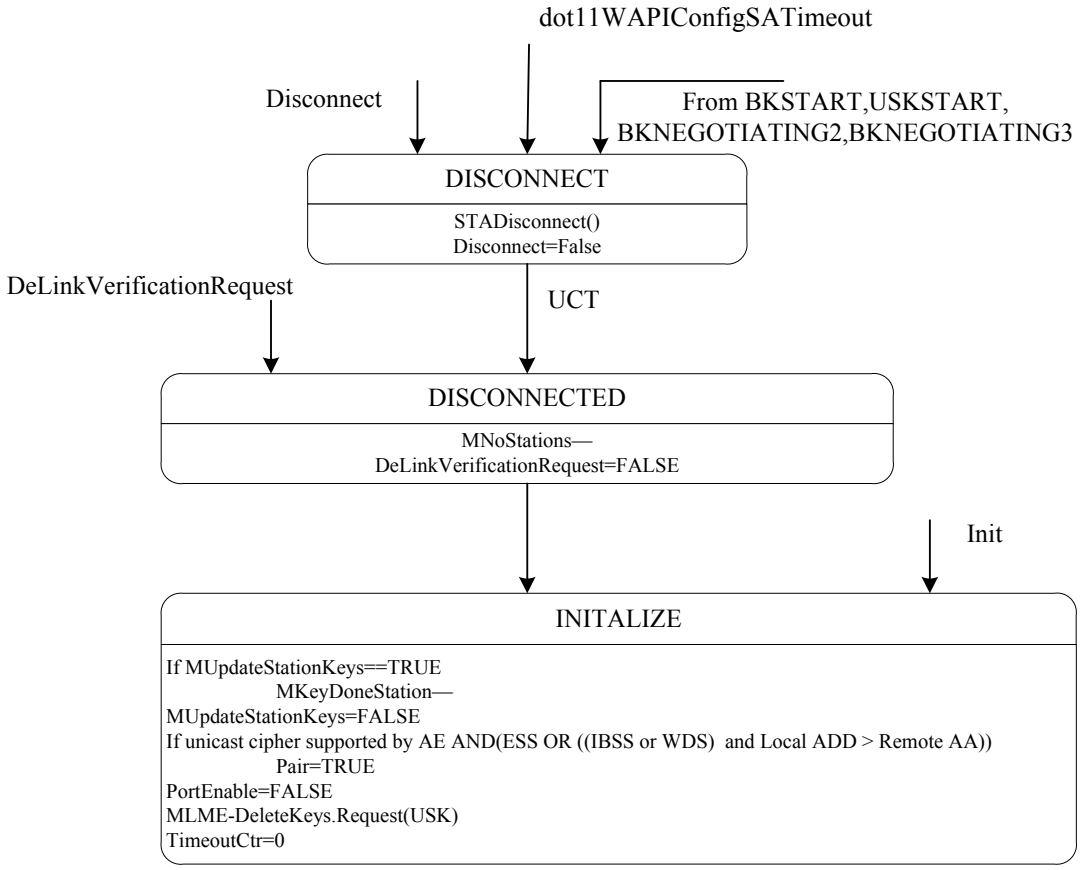

Figure 43au—Authenticator state machine, Part 3

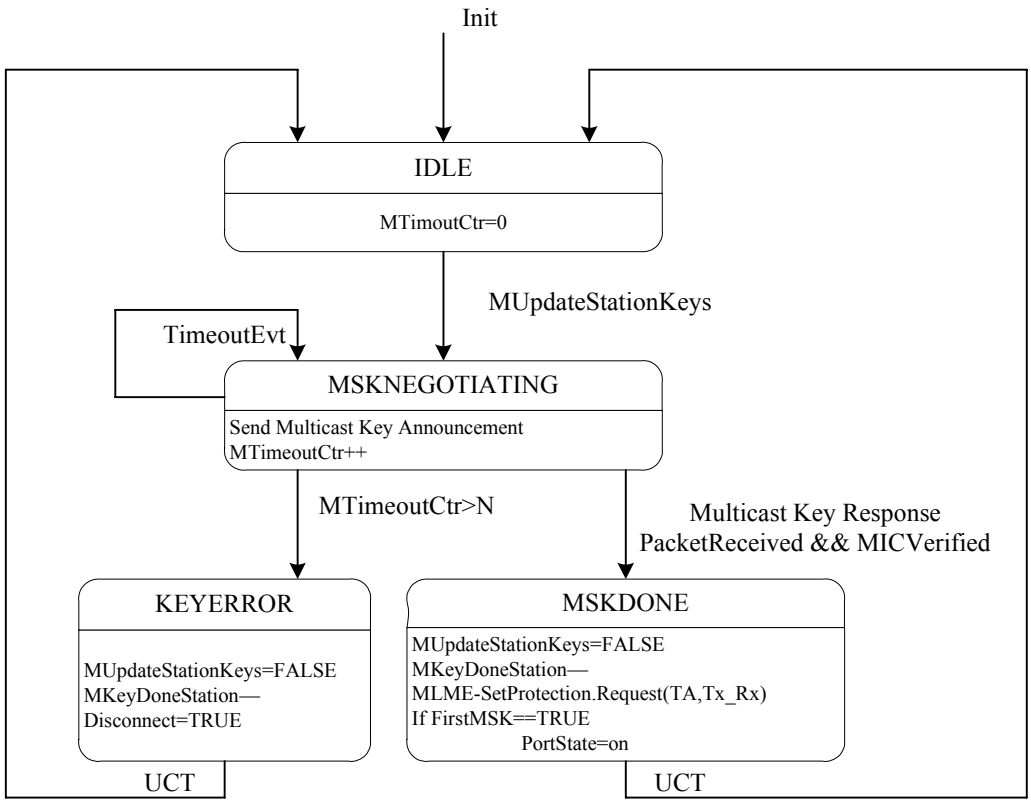

Figure 43av—Authenticator state machine, Part 4

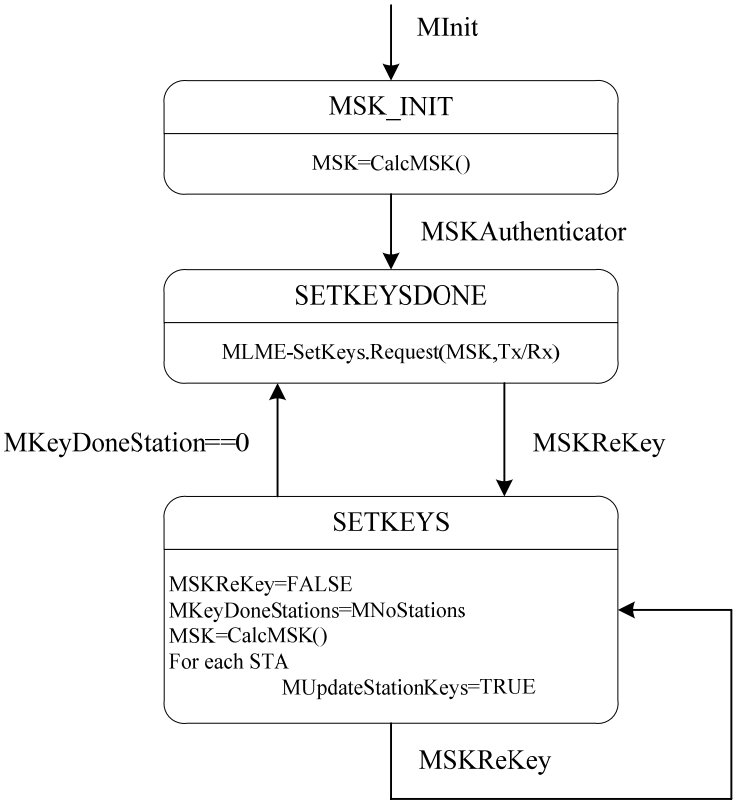

Figure43aw—Authenticator state machine, Part 5

8.3.2.1 Authenticator state machine states

8.3.2.1.1 Authenticator state machine: Certificate Authentication and USK Negotiation (per STA)

The following list summarizes the states the Authenticator state machine uses to support the Certificate Authentication and the USK Negotiation:

— **AUTHENTICATION**: This state is entered when an AuthenticationRequest is sent from the management entity to authenticate a BSSID.

— **BKSTART:** This state is entered from AUTHENTICATION to start the BK Negotiation or if BKReKey is set TRUE.

— **INITPSK:** This state is entered when a PSK is configured.

— **BKNEGOTIATING:** This state is entered when the Access Authentication Request packet is received.

— **BKNEGOTIATING2:** This state is entered when the certificate is required to be verified.

— **BKNEGOTIATING3:** This state is entered when the Certificate Authentication Response packet is received.

— **BKDONE:** This state is entered when the certificate is valid and the signature for Access Authentication Request packet is verified.

— **USKSTART:** This state is entered from BKDONE to start the USK Negotiation when BKAvailable is set TRUE or if USKReKey is set TRUE.

— **USKNEGOTIATING:** This state is entered when the Unicast Key Negotiation Response packet is received.

— **USKDONE:** This state is entered when the HMAC for the Unicast Key Negotiation Response packet is verified.

— **DISCONNECT**: This state is entered if Disconnect is set TRUE or WAI procedure is timeout. It sends a Deauthentication message to the STA and enters the INITIALIZE state.

— **DISCONNECTED**: This state is entered when Disassociation or Deauthentication messages are received.

— **INITIALIZE**: This state is entered from the DISCONNECTED state, when a Deauthentication request event occurs, or when the station initializes. The state initializes the key state variables.

8.3.2.1.2 Authenticator state machine: Multicast Key Announcement (per STA)

The following list summarizes the states the Authenticator state machine uses to support the Multicast Key Announcement:

— **IDLE:** This state is entered when no Multicast Key Announcement is occurring.

— **KEYERROR:** This state is entered if the Multicast Key Response packet for the Multicast Key Announcement is not received.

— **MSKDONE:** This state is entered when the Multicast Key Response packet for the Multicast Key Announcement is received from the Supplicant.

— **MSKNEGOTIATING:** This state is entered when the MSK is to be sent to the Supplicant.

8.3.2.1.3 Authenticator state machine: Multicast Key Announcement (global)

The following list summarizes the states the Authenticator state machine uses to coordinate a multicast key update of all STAs:

— **MSK_INIT:** This state is entered on system initialization.

— **SETKEYS:** This state is entered if the MSK is to be updated on all Supplicants.

— **SETKEYSDONE:** This state is entered if the MSK has been updated on all Supplicants.

8.3.2.2 Authenticator state machine variables

The following list summarizes the variables used by the Authenticator state machine:

— *AuthenticationRequest* – This variable is set to TRUE if the STA's ISO/IEC 8802.11 management entity requires an association to be authenticated. This can be set when the STA associates or at other time.

— *DeauthenticationRequest* – This variable is set to TRUE if a Disassociation or Deauthentication message is received.

— *Disconnect* – This variable is set to TRUE when the STA should initiate a deauthentication.

— *PacketReceived* – This variable is set to TRUE when a WAI packet is received.

— *MTimeoutCtr* – This variable maintains the count of a specific WAI packet receive timeouts for the Multicast Key Announcement. It is incremented each time a timeout occurs on a specific WAI packet receive event and is initialized to 0.

— *MInit* – This variable is used to initialize the multicast key state machine.

— *Init* – This variable is used to initialize per-STA state machine.

— *TimeoutEvt* – This variable is set to TRUE if the WAI packet sent out fails to obtain a response from the Supplicant. The variable may be set by management action or set by the operation of a timeout while in the BKSTART, USKSTART, BKNEGOTIATING2 and MSKNEGOTIATING states.

— *TimeoutCtr* **–** This variable maintains the count of WAI packet receive timeouts. It is incremented each time a timeout occurs on a WAI packet receive event and is initialized to 0.

— *HMACVerified* – This variable is set to TRUE if the MIC on the received WAI packet is verified and is correct. Any WAI packets with an invalid HMAC will be dropped and ignored.

— *MSKAuthenticator* – This variable is set to TRUE if the Authenticator is on an AP or it is the designated Authenticator for an IBSS.

— *MKeyDoneStations* – Count of number of STAs left to have their MSK updated. This is a global variable.

— *MSKRekey* – This variable is set to TRUE when a Multicast Key Announcement is required. This is a global variable.

— *BKReKey* – This variable is set to TRUE when a BK rekeying is required.

— *USKReKey* – This variable is set to TRUE when a USK rekeying is required.

— *MUpdateStationKeys* – This variable is set to TRUE when a new MSK is available to be sent to Supplicants.

— *MNoStations* – This variable counts the number of Authenticators so it is known how many Supplicants need to be sent the MSK. This is a global variable.

— *USK* – This variable is the current USK.

— *MSK* – This variable is the current MSK.

— *BK* – This variable is the buffer holding the current BK.

—*Certificate_Verificaton*– This variable is set to TRUE when the certificate is required to be verified.

—PortControlled – This variable is the methods by which the port is controlled. The value is Auto or ForceUnauthenticated.
—Portenabled – This variable is set to TRUE if the port control mechanism is applied.
—PortState – This variable is set to on if the data frame is permitted to pass through the port.
—Certificate Authentication– This variable is set to TRUE if the certificate authentication and key management is chosen.
—PSK Authentication– This variable is set to TRUE if the PSK authentication and key management is chosen.

8.3.2.3 Authenticator state machine procedures

The following list summarizes the procedures used by the Authenticator state machine
— **STADisconnect**() – Execution of this procedure Deauthenticates the STA.
— **CalcMSK**() – Generates the MSK.

9. MAC sublayer functional description

See 9 of ISO/IEC 8802-11:2006.

10. Layer management

10.1 Overview of management model

See 10.1 of ISO/IEC 8802-11:2006.

10.2 Generic management primitives

See10.2 of ISO/IEC 8802-11:2006.

10.3 MLME SAP interface

10.3.1 Power management

See 10.3.1 of ISO/IEC 8802-11:2006.

10.3.2 Scan

10.3.2.1 MLME-SCAN.request

See I10.3.2.1 of SO/IEC 8802-11:2006;

10.3.2.2 MLME-SCAN.confirm

10.3.2.2.1 Function

See 10.3.2.2.1 of ISO/IEC 8802-11:2006.

10.3.2.2.2 Semantics of the service primitive

The primitive parameters are as follows：

MLME- SCAN_.confirm (

BSSDescriptionSet

ResultCode

WAPI

)

Name	Type	Valid range	Description
BSSDescriptionSet	Set of BSSDescriptions	N/A	The BSSDescriptionSet is returned to indicate the results of the scan request. It is a set containing zero or more instances of a BSSDescription
ResultCode	Enumeration	SUCCESS, INVALID_ PARAMETERS	Indicates the result of the MLMESCAN. confirm
WAPI	WAPI Information Element	As defined in frame Format	A description of the cipher suites and AKM suites supported in the BSS

Each BSSDescription consists of the following elements:

Name	Type	Valid range	Description
BSSID	MACAddress	N/A	The BSSID of the found BSS
SSID	Octet string	1–32 octets	The SSID of the found BSS
BSSType	Enumeration	INFRASTRUCTURE, INDEPENDENT	The type of the found BSS
Beacon Period	Integer	N/A	The Beacon period of the found BSS (in TU)
DTIM Period	Integer	As defined in frame format	The DTIM period of the BSS (in beacon periods)
Timestamp	Integer	N/A	The timestamp of the received frame (probe response/beacon) from the found BSS
Local Time	Integer	N/A	The value of the STA’s TSF timer at the start of reception of the first octet of the timestamp field of the received frame (probe response or beacon) from the found BSS

Name	Type	Valid range	Description
PHY parameter set	As defined in frame format	As defined in frame format	The parameter set relevant to the PHY
CF parameter set	As defined in frame format	As defined in frame format	The parameter set for the CF periods, if found BSS supports CF mode
IBSS parameter set	As defined in frame format	As defined in frame format	The parameter set for the IBSS, if found BSS is an IBSS
CapabilityInformation	As defined in frame format	As defined in frame format	The advertised capabilities of the BSS
BSSBasicRateSet	Set of integers	2–127 inclusive (for each integer in the set)	The set of data rates (in units of 500 kb/s) that must be supported by all STAs that desires to join this BSS. The STAs must be able to receive at each of the data rates listed in the set

10.3.2.2.3 When generated

See **10.3.2.2.3 of** ISO/IEC 8802-11:2006.

10.3.2.2.4 Effect of receipt

See **10.3.2.2.4 of** ISO/IEC 8802-11:2006.

10.3.3 Synchronization

See 10.3.3 of ISO/IEC 8802-11:2006.

10.3.4 Authenticate

This mechanism supports the process of establishing an authentication relationship with a peer MAC entity.

10.3.4.1 MLME-AUTHENTICATE .request

10.3.4.1.1 Function

This primitive requests an authentication with a specified peer MAC entity.

10.3.4.1.2 Semantics of the service primitive

The primitive parameters are as follows:

```
MLME- AUTHENTICATE_.request        (
                                        PeerSTAAddress,
                                        AuthenticateFailureTimeout
                                        )
```

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid Individual MAC address	Specifies the address of the peer MAC entity with which to perform the Authentication process
AuthenticationFailure-Timeout	Integer	≥ 1	Specifies a time limit (in TU) after which the authentication procedure will be terminated

10.3.4.1.3 When generated

This primitive is generated by the SME for a STA to establish authentication with a specified peer MAC entity in order to permit Class 2 frames to be exchanged between the two STAs. During the authentication procedure, the SME may generate additional MLME--AUTHENTICATE.request primitives.

10.3.4.1.4 Effect of request

This primitive initiates an authentication procedure. The MLME subsequently issues an MLME-AUTHENTICATE.confirm that reflects the results.

10.3.4.2 MLME- AUTHENTICATE.confirm

10.3.4.2.1 Function

This primitive reports the results of an authentication attempt with a specified peer MAC entity.

10.3.4.2.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME- AUTHENTICATE.confirm (
PeerSTAAddress,
ResultCode
)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC entity with which the authentication process was attempted. This value must match the peerSTAAddress parameter specified in the corresponding

Name	Type	Valid range	Description
			MLME-AUTHENTICATE_equest
ResultCode	Enumeration	SUCCESS INVALID_PARAMETERS TIMEOUT TOO_MANY_ SIMULTANEOUS_ REQUESTS REFUSED	Indicates the result of the MLME-AUTHENTICATE_.request

10.3.4.2.3 When generated

This primitive is generated by the MLME as a result of an MLME-–AUTHENTICATE.request to perform the authentication with a specified peer MAC entity.

10.3.4.2.4 Effect of receipt

The SME is notified of the results of the authentication procedure.

10.3.4.3 MLME- AUTHENTICATE.indication

10.3.4.3.1 Function

This primitive reports the establishment of an authentication relationship with a specified peer MAC entity.

10.3.4.3.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME--AUTHENTICATE.indication (
PeerSTAAddress
)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC address with which the authentication relationship was established

10.3.4.3.3 When generated

This primitive is generated by MLME as a result of the establishment of an authentication relationship with a specified peer MAC entity that resulted from an authentication procedure that is initiated by that specified peer MAC entity.

10.3.4.3.4 Effect of receipt

The SME is notified of the establishment of the authentication relationship.

10.3.5 DEAUTHENTICATE

This mechanism supports the process of invalidating an authentication relationship with a peer MAC entity.

10.3.5.1 MLME- DEAUTHENTICATE.request

10.3.5.1.1 Function

This primitive requests that the authentication relationship with a specified peer MAC entity be invalidated.

10.3.5.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME- **DEAUTHENTICATE**.request (
PeerSTAAddress,
)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC entity with which to invalidate the relationship of the authentication

10.3.5.1.3 When generated

This primitive is generated by the SME for a STA to invalidate the authentication with a specified peer MAC entity in order to prevent the exchange of Class 2 frames between the two STAs. During the invalidation procedure, the SME may generate additional MLME- DEAUTHENTICATE.request primitives.

10.3.5.1.4 Effect of receipt

This primitive initiates the invalidation procedure of the authentication. The MLME subsequently issues an MLME--DEAUTHENTICATE .confirm that reflects the results.

10.3.5.2 MLME- DEAUTHENTICATE.confirm

10.3.5.2.1 Function

This primitive reports the result of a Deauthentication attempt with a specified peer MAC entity.

10.3.5.2.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME--DEAUTHENTICATE.confirm (
PeerSTAAddress,
ResultCode
)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC entity with which the invalidation process of the authentication was attempted
ResultCode	Enumeration	SUCCESS INVALID_PARAMETERS TOO_MENY_ SIMULTANEOUS_ REQUESTS	Indicates the result of the MLME--DEAUTHENTICATE.request

10.3.5.2.3 When generated

This primitive is generated by the MLME as the result of an MLME--DEAUTHENTICATE.request to invalidate the authentication relationship with a specified peer MAC entity.

10.3.5.2.4 Effect of receipt

The SME is notified of the results of the authentication procedure.

10.3.5.3 MLME--DEAUTHENTICATE.indication

10.3.5.3.1 Function

This primitive reports the invalidation of an authentication relationship with a specified peer MAC entity.

10.3.5.3.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME- DEAUTHENTICATE.indication (
PeerSTAAddress,

ReasonCode

)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC address with which the authentication relationship is invalidated
ReasonCode	As defined in frame format	As defined in frame format	Specifies the reason the deauthentication procedure is initiated

10.3.5.3.3 When generated

This primitive is generated by the MLME as a result of the invalidation of the authentication relationship with specified peer MAC entity.

10.3.5.3.4 Effect of receipt

The SME is notified of the invalidation of the specific authentication relationship.

10.3.6 Associate

The following primitives describe how a STA becomes associated with an AP.

10.3.6.1 MLME-ASSOCIATE.request

10.3.6.1.1 Function

See **10.3.6.1.1 of ISO/IEC 8802-11:2006.**

10.3.6.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-ASSOCIATE.request(
PeerSTAAddress,
AssociateFailureTimeout,
CapabilityInformation,
ListenInterval,
Supported Channels
WAPI
)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC entity with which to perform the association process

Name	Type	Valid range	Description
AssociateFailureTimeout	Integer	≥1	Specifies a time limit (in TU) after which the associate procedure will be terminated
CapabilityInformation	As defined in frame format	As defined in frame format	Specifies the operational capability definitions to be used by the MAC entity
ListenInterval	Integer	≥0	Specifies the number of beacon intervals that may pass before the STA awakens and listens for the next beacon
WAPI	WAPI Information element	As defined in frame format	Specifies the cipher suites and the AKM suites that BSS supports

10.3.6.1.3 When generated

See 10.3.6.1.3 of ISO/IEC 8802-11:2006.

10.3.6.1.4 Effect of receipt

See 10.3.6.1.4 of ISO/IEC 8802-11:2006.

10.3.6.2 MLME-ASSOCIATE.confirm

See 10.3.6.2 of ISO/IEC 8802-11:2006.

10.3.6.3 MLME-ASSOCIATE.indication

10.3.6.3.1 Function

See 10.3.6.3.1 of ISO/IEC 8802-11:2006.

10.3.6.3.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-ASSOCIATE.indication(
PeerSTAAddress,
WAPI
)

Name	Type	Valid range	Description
PeerSTAAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC entity with which the association was established

Name	Type	Valid range	Description
WAPI	WAPI Information Element	As defined in frame format	Specifies the cipher suites and the AKM suites that BSS supports. There is only one unicast cipher suite in the WAPI information element

10.3.6.3.3 When generated

See 10.3.6.3.3 of ISO/IEC 8802-11:2006.

10.3.6.3.4 Effect of receipt

See 10.3.6.3.4 of ISO/IEC 8802-11:2006.

10.3.7 Reassociate

The following primitive requests a change in association to a specified new peer MAC entity that is acting as an AP.

10.3.7.1 MLME-REASSOCIATE.request

10.3.7.1.1 Function

See 10.3.7.1.1 of ISO/IEC 8802-11:2006.

10.3.7.1.2 Semantics of the service primitive

The primitive parameters are as follows:

```
MLME-REASSOCIATE.request(
                        NewAPAddress,
                        ReassociateFailureTimeout,
                        CapabilityInformation,
                        ListenInterval,
                        Supported Channels,
                        WAPI
                         )
```

Name	Type	Valid range	Description
NewAPAddress	MACAddress	Any valid individual MAC address	Specifies the address of the peer MAC entity with which to perform the reassociation process
ReassociateFailureTimeout	Integer	≥1	Specifies a time limit (in TU) after which the reassociate procedure will be terminated

CapabilityInformation	As defined in frame format	As defined in frame format	Specifies the operational capability definitions to be used by the MAC entity
ListenInterval	Integer	≥0	Specifies the number of beacon intervals that may pass before the STA awakens and listens for the next beacon.
WAPI	WAPI Information element	As defined in frame format	Specifies the cipher suites and the AKM suites that BSS supports

10.3.7.1.3 When generated

See 10.3.7.1.3 of ISO/IEC 8802-11:2006.

10.3.7.1.4 Effect of receipt

See 10.3.7.1.4 of ISO/IEC 8802-11:2006.

10.3.7.2 MLME-REASSOCIATE.confirm

See 10.3.7.2 of ISO/IEC 8802-11:2006.

10.3.7.3 MLME-REASSOCIATE.indication

10.3.7.3.1 Function

See 10.3.7.3.1 of ISO/IEC 8802-11:2006.

10.3.7.3.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-REASSOCIATE.indication(
PeerSTAAddress,
WAPI
)

Name	Type	Valid range	Description
ResultCode	Enumeration	SUCCESS, INVALID_ PARAMETERS, TIMEOUT, REFUSED	Indicates the result of the MLME-REASSOCIATE.request
WAPI	WAPI Information Element	As defined in frame format	Specifies the cipher suites and the AKM suites that BSS supports. There is only one unicast cipher

			suite in the WAPI information element.

10.3.7.3.3 When generated

See 10.3.7.3.3 of ISO/IEC 8802-11:2006.

10.3.7.3.4 Effect of receipt

See 10.3.7.3.4 of ISO/IEC 8802-11:2006.

10.3.8 Disassociate

See 10.3.8 of ISO/IEC 8802-11:2006.

10.3.9 Reset

See 10.3.9 of ISO/IEC 8802-11:2006.

10.3.10 Start

See 10.3.10 of ISO/IEC 8802-11:2006.

10.3.17 MLME-SETWPIKEYS

This primitive sets the keys used by the STA and the peer STA authenticated when WPI is enabled.

10.3.17.1 MLME-SETWPIKEYS.request

10.3.17.1.1 Function

This primitive requests a STA's MAC entity set the keys used by the STA and another peer STA's MAC entity when WPI is enabled.

10.3.17.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-SETWPIKEYS.request (
Keylist
)

Name	Type	Description

Keylist	A set of SetKeyDescriptors	The list of keys to be used by the MAC.

Each SetKeyDescriptor consists of the following elements:

Name	Type	Description
Key	Octet string	Data encryption key and data integrity key
Length	Integer	The length of the key in octet
KeyIdx	Integer	The number of the key, with a value of 0 or 1
KeyType	Integer	Multicast, Unicast,STAKey
PeerSTAAddress	MACAddress	Any valid individual MAC address. This parameter is valid only when the Key Type value is Unicast, when the Key Type value is Multicast and the STA is in an IBSS, or when the Key Type value is STAKey.
Authenticator/Supplicant or Initiator/Peer	Boolean	Whether the key is configured by the Authenticator(Initiator) or Supplicant(Peer); true indicates Authenticator or Initiator; false indicates Supplicant or Peer.
GSN	Integer	The sequence number of the currently encrypted multicast package. It is valid only when KeyType is multicast key.
Cipher Suite Selector	4 octets	As defined in the WAPI information element format The cipher suite required for this association.

10.3.17.1.3 When generated

This primitive is generated by the SME at any time when the key negotiation is completed or the shared key is set.

10.3.17.1.4 Effect of receipt

This primitive enables MAC to perform the privacy of the data using the key. The MLME subsequently issues an MLME-SETWPIKEYS.confirm that reflects the results.

10.3.17.2 MLME-SETWPIKEYS.confirm

10.3.17.2.1 Function

This primitive confirms that the action of the associated MLME-SETWPIKEYS.request primitive has been

completed.

10.3.17.2.2 Semantics of the service primitive

This primitive has no parameters.

10.3.17.2.3 When generated

This primitive is generated by the MAC in response to receipt of an MLME-SETWPIKEYS.request primitive.This primitive is issued when the action requested has been completed.

10.3.17.2.4 Effect of receipt

The SME is notified that the requested action of the MLME-SETWPIKEYS.request primitive is completed.

10.3.18 MLME-DELETEWPIKEYS

The following primitives delete the keys that a STA used in WPI.

10.3.18.1 MLME-DELETEWPIKEYS.request

10.3.18.1.1 Function

This primitive requests a STA's MAC entity delete the keys that the STA used in WPI with the peer STA's MAC entity when WPI is enabled.

10.3.18.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-DELETEWPIKEYS.request (
Keylist
)

Name	Type	Description
Keylist	A set of DeleteKeyDescriptors	The list of keys to be used by the MAC.

Each DeleteKeyDescriptor consists of the following elements:

Name	Type	Description

PeerSTAAddress	MACAddress	Any valid individual MAC address. This parameter is valid only when the Key Type value is Unicast, when the Key Type value is Multicast and the STA is in an IBSS, or when the Key Type value is STAKey.
KeyIdx	Integer	The number of the key, with a value of 0 or 1
KeyType	Integer	Multicast, Unicast, STAKey

10.3.18.1.3 When generated

This primitive is generated by the SME at any time when a STA wishes to delete the key.

10.3.18.1.4 Effect of receipt

This primitive disables MAC to perform the privacy of the data using the key. The MLME subsequently issues an MLME- DELETEWPIKEYS.confirm that reflects the results.

10.3.18.2 MLME-DELETEWPIKEYS.confirm

10.3.18.2.1 Function

This primitive confirms that the action of the associated MLME-DELETEWPIKEYS.request primitive has been completed.

10.3.18.2.2 Semantics of the service primitive

This primitive has no parameters.

10.3.18.2.3 When generated

This primitive is generated by the MAC in response to receipt of an MLME-DELETEWPIKEYS.request primitive. This primitive is issued when the action requested has been completed.

10.3.18.2.4 Effect of receipt

The SME is notified that the requested action of the MLME-DELETEWPIKEYS.request primitive is completed.

10.3.19 MLME-STAKEYESTABLISHED

10.3.19.1 MLME-STAKEYESTABLISHED.indication

10.3.19.1.1 Function

This primitive notifies the SME that a STAKey is required.

10.3.19.1.2 Semantics of the service primitive

This primitive has two parameters, the MAC addresses of the two STAs.
The primitive parameters are as follows:

MLME-STAKEYESTABLISHED.indication (
Address1,
Address2
)

Name	Type	Valid range	Description
Address1	MACAddress	Any valid individual MAC address	MAC address of initiating STA.
Address2	MACAddress	Any valid individual MAC address	MAC address of peer STA.

10.3.19.1.3 When generated

This primitive is generated by the MAC when a STAKey is required.

10.3.19.1.4 Effect of receipt

The SME is notified that a STAKey is required, is made aware of the two STA MAC addresses involved, and can then send STAKey Handshake messages to each STA.

10.3.20 SetProtection

10.3.20.1 MLME-SETPROTECTION.request

10.3.20.1.1 Function

This primitive indicates whether protection is required for frames sent to and received from the indicated MAC address.

10.3.20.1.2 Semantics of the service primitive

The primitive parameters are as follows:

MLME-SETPROTECTION.request(

Protectlist

)

Name	Type	Description
Protectlist	A set of protection elements	The list of how each key is being used currently.

Each Protectlist consists of the following elements:

Name	Type	Description
Address	MACAddress	Any valid individual MAC address. This parameter is valid only when the Key Type value is Unicast or STAKey or when the Key Type value is Multicast and is from an IBSS STA.
ProtectType	Enumeration (None, Rx, Tx, Rx_Tx)	The protection value for this MAC.
KeyType	Integer	Defines whether this key is a multicast key, unicast key, or STAKey.

10.3.20.1.3 When generated

This primitive is generated by the SME when protection is required for frames sent to and received from the indicated MAC address.

10.3.20.1.4 Effect of receipt

Receipt of this primitive cause the MAC to set the protection and to protect data frames as indicated in the ProtectType element of the Protectlist parameter:

— None: Specifies that data frames neither from the MAC address nor to the MAC address shall be protected.
— Rx: Specifies that data frames from MAC address shall be protected.
— Tx: Specifies that data frames to MAC address shall be protected.
— Rx_Tx: Specifies that data frames to and from MAC address shall be protected.

Once it is specified that a data frame is protected to or from a MAC address, this shall be reset by the MLME-SETPROTECTION.request primitive. The MLME-SETPROTECTION.request primitive deletes the state by specifying None.

10.3.20.2 MLME-SETPROTECTION.confirm

10.3.20.2.1 Function

This primitive indicates that the frame protection request is completed.

10.3.20.2.2 Semantics of the service primitive

There are no parameters for this primitive.

10.3.20.2.3 When generated

This primitive is generated by the MAC when the protection request is completed.

10.3.20.2.4 Effect of receipt

The SME is notified that the protection request is completed.

10.3.21 MLME-PROTECTEDFRAMEDROPPED

10.3.21.1 MLME- PROTECTEDFRAMEDROPPED.indication

10.3.21.1.1 Function

This primitive notifies the SME that a frame has been dropped because a temporal key was unavailable.

10.3.21.1.2 Semantics of the service primitive

This primitive has two parameters, the MAC addresses of the two STAs.
The primitive parameters are as follows:

MLME- PROTECTEDFRAMEDROPPED.indication (
Address1,
Address2
)

Name	Type	Valid range	Description
Address1	MACAddress	Any valid individual MAC address	MAC address of SA.
Address2	MACAddress	Any valid individual MAC address	MAC address of RA.

10.3.21.1.3 When generated

This primitive is generated by the MAC when a frame is dropped because no temporal key is available for the frame.

10.3.21.1.4 Effect of receipt

The SME is notified that a frame was dropped. The SME can use this information in an IBSS to initiate a

security association to the peer STA.

10.4 PLME SAP interface

See 10.4 of ISO/IEC 8802-11:2006.

11. MAC sublayer management entity

11.1 Synchronization

See 11.1 of ISO/IEC 8802-11:2006.

11.2 Power management

See 11.2 of ISO/IEC 8802-11:2006.

11.3 Association and reassociation

This subclause describes the procedures used for ISO/IEC 8802.11 authentication and deauthentication. The states used in this description are those defined in 5.5.

11.3.1 Authentication—originating STA

Upon receipt of an MLME-AUTHENTICATE.request primitive, the originating STA shall authenticate with the indicated STA using the following procedure:

a) In an ESS, or optionally in an IBSS, the STA shall execute the authentication mechanism.
b) If the authentication is successful, the state variable of the indicated STA shall be set to State 2.
c) The STA shall issue an MLME-AUTHENTICATE.confirm primitive to inform the SME of the result of the authentication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive before invoking MLME-AUTHENTICATE.request primitive.

11.3.2 Authentication—destination STA

Upon receipt of an authentication frame with authentication transaction sequence number equal to 1, the destination STA shallauthenticate with the indicated STA using the following procedure:

a) The STA shall execute the authentication mechanism.
b) The STA shall issue an MLME-AUTHENTICATE.indication primitive to inform the SME of the

authentication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using the MLME-DELETEWPIKEYS.request primitive upon receiving an MLME-AUTHENTICATE.indication primitive.

If the STA is in an IBSS, if the SME decides to initiate a WAPI association, and if the SME does not know the security policy of the peer, it may issue a unicast Probe Request frame to the peer by invoking an MLME-SCAN.request to discover the peer's security policy.

11.3.3 Deauthentication—originating STA

Upon receipt of an MLME-DEAUTHENTICATE.request primitive, the originating STA shall deauthenticate with the indicated STA using the following procedure:

a) If the state variable of the indicated STA is in State 2 or State 3, the STA shall send a Deauthentication frame to the indicated STA.
b) The state variable of the indicated STA shall be set to State 1.
c) The STA shall issue an MLME-DEAUTHENTICATE.confirm primitive to inform the SME of the completion of the Deauthentication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using an MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION. request(None) before invoking an MLME-DEAUTHENTICATE.request primitive.

11.3.4 Deauthentication—destination STA

Upon receipt of a Deauthentication frame, the destination STA shall deauthenticate with the indicated STA using the following procedure:

a) The state variable of the indicated STA shall be set to State 1.
b) The STA shall issue an MLME-DEAUTHENTICATE.indication primitive to inform the SME of the deauthentication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using the MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION.request (None) upon receiving an MLME-DEAUTHENTICATE.indication primitive.

11.4 Association, reassociation, and disassociation

This subclause defines how a STA associates and reassociates with an AP and how it disassociates from it. The states used in this description are those defined in 5.5.

11.4.1 STA association procedures

Upon receipt of an MLME-ASSOCIATE.request primitive, a STA shall associate with an AP via the following procedure:

a) The STA shall transmit an Association Request frame to the AP with which that STA is authenticated. If the MLME-ASSOCIATE.request primitive contained a WAPI information element with only one unicast cipher suite and only one authenticated key suite, this WAPI information element shall be included in the Association Request frame.
b) If an Association Response frame is received with a status value of "successful," the STA is now associated with the AP. The state variable shall be set to State 3, and the MLME shall issue an MLME-ASSOCIATE.confirm primitive indicating the successful completion of the operation.
c) If an Association Response frame is received with a status value other than "successful" or the AssociateFailureTimeout expires, the STA is not associated with the AP. The MLME shall issue an MLME-ASSOCIATE.confirm primitive indicating the failure of the operation.
d) The SME shall establish a WAPI association by calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive before invoking MLME-ASSOCIATE.request primitive.

11.4.2 AP association procedures

When an Association Request frame is received from a STA, the AP shall associate with the STA using the following procedure:

a) If the STA is not authenticated, the AP shall transmit a Deauthentication frame to the STA and terminate the association procedure.
b) In a WAPI association, the AP shall check if the values received in the WAPI information element match the AP's security policy. If not, the association shall not be accepted.
c) The AP shall transmit an Association Response. If the status value is "successful," the association identifier assigned to the STA shall be included in the response.
d) When the Association Response with a status value of "successful" is acknowledged by the STA, the STA is considered to be associated with this AP. The state variable for the STA shall be set to State 3.
e) The MLME shall issue an MLME-ASSOCIATE.indication primitive to inform the SME of the association.
f) The SME shall establish a WAPI association by calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.
g) The SME will inform the DS of the new association.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive upon receiving an MLME-ASSOCIATE.indication primitive.

11.4.3 STA reassociation procedures

Upon receipt of an MLME-REASSOCIATE.request primitive, a STA shall reassociate with an AP via the following procedure:

a) If the state variable is in State 1, the STA shall inform the SME of the failure of the reassociation by issuing an MLME-REASSOCIATE.confirm primitive.
b) The STA shall transmit a Reassociation Request frame to the new AP. If the MLME-REASSOCIATE.request primitive contained a WAPI information element with only one unicast cipher suite and only one authenticated key suite, this WAPI information element shall be included in the Reassociation Request frame.
c) If a Reassociation Response frame is received with a status value of "successful," the STA is now associated with the new AP. The state variable shall be set to State 3, and the MLME shall issue an MLME-REASSOCIATE.confirm primitive indicating the successful completion of the operation.
d) If a Reassociation Response frame is received with a status value other than "successful" or the AssociateFailureTimeout expires, the STA is not associated with the AP. The MLME shall issue an MLME-REASSOCIATE.confirm primitive indicating the failure of the operation.
e) The SME shall establish a WAPI association calling MLME.SETPROTECTION.request primitive with ProtectType set to "Rx_Tx," or it shall do nothing if it does not wish to secure communication.

The STA's SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive before invoking MLME-REASSOCIATE.request primitive.

11.4.4 AP reassociation procedures

Whenever a Reassociation Request frame is received from a STA, the AP uses the following procedure to support reassociation:

a) If the STA is not authenticated, the AP shall transmit a Deauthentication frame to the STA and terminate the reassociation procedure.
b) In a WAPI association, the AP shall check if the values received in the WAPI information element match the AP's security policy. If not, the association shall not be accepted.
c) The AP shall transmit a Reassociation Response frame. If the status value is "successful," the association identifier assigned to the STA shall be included in the response.
d) When the Reassociation Response frame with a status value of "successful" is acknowledged by the STA, the STA is considered to be associated with this AP. The state variable of the STA shall be set to State 3.
e) The MLME shall issue an MLME-REASSOCIATE.indication primitive to inform the SME of the

association.

f) The SME shall establish a WAPI association calling MLME.SETPROTECTION.request primitive with ProtectType set to “Rx_Tx,” or it shall do nothing if it does not wish to secure communication.

g) The SME will inform the DS of the new association.

The STA’s SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using MLME-DELETEWPIKEYS.request primitive upon receiving an MLME-REASSOCIATE.indication primitive.

11.4.5 STA disassociation procedures

Upon receipt of an MLME-DISASSOCIATE.request primitive, an associated STA shall disassociate from an AP using the following procedure:

a) The STA shall transmit a Disassociation frame to the AP with which that STA is associated.
b) The state variable of the AP shall be set to State 2 if and only if it was not State 1.
c) The MLME shall issue an MLME-DISASSOCIATE.confirm primitive indicating the successful completion of the operation.

The STA’s SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using the MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION.request (None) before invoking an MLME-DISASSOCIATE.request primitive.

11.4.6 AP disassociation procedures

Upon receipt of a Disassociation frame from an associated STA, the AP shall disassociate the STA via the following procedure:

a) The state variable of the STA shall be set to State 2.
b) The MLME shall issue an MLME-DISASSOCIATE.indication primitive to inform the SME of the disassociation.
c) The SME will update the DS.

The STA’s SME shall delete any USKSA and temporal keys held for communication with the indicated STA by using an MLME-DELETEWPIKEYS.request primitive and by invoking MLME-SETPROTECTION. request (None) upon receiving an MLME-DISASSOCIATE.indication primitive.

12. PHY service specification

See 12 of ISO/IEC 8802-11:2006.

13. PHY management

See 13 of ISO/IEC 8802-11:2006.

14. Frequency-Hopping spread spectrum (FHSS) PHY specification for the 2.4 GHz industrial, scientific, and medical (ISM) band

See 14 of ISO/IEC 8802-11:2006.

15. DSSS PHY specification for the 2.4 GHz band designated for ISM applications

See 15 of ISO/IEC 8802-11:2006.

16. Infrared (IR) PHY specification

See 16 of ISO/IEC 8802-11:2006.

17. Orthogonal frequency division multiplexing (OFDM) PHY specification for the 5 GHz band

See 17 of ISO/IEC 8802-11:2006.

18. High Rate direct sequence spread spectrum (HR/DSSS) PHY specification

See 18 of ISO/IEC 8802-11:2006.

Annex A

(normative)

Protocol Implementation Conformance Statements (PICS)

A.1 Introduction

See annex A.1 of ISO/IEC 8802-11:2006.

A.2 Abbreviations and special symbols

See annex A.2 of ISO/IEC 8802-11:2006.

A.3 Instructions for completing the PICS proforma

See annex A.1 of ISO/IEC 8802-11:2006.

A.4 PICS proforma

A.4.1 Implementation identification

See annex A.1 of ISO/IEC 8802-11:2006.

A.4.2 Protocol summary

See annex A.1 of ISO/IEC 8802-11:2006.

A.4.3 IUT configuration

See annex A.1 of ISO/IEC 8802-11:2006.

A.4.4 MAC protocol

A.4.4.1 MAC protocol capabilities

Item	Protocol capability	References	Status	Support
	Are the following MAC protocol capabilities supported?			
PC1	Authentication service	5.4.3.1, 5.4.3.2, 5.7.6, 5.7.7, Annex C	M	Yes □ No □
PC1.1	Authentication state	5.55.7.6	M	Yes □ No □
PC1.2	Open System Authentication		M	Yes □ No □

Item	Protocol capability	References	Status	Support
PC2	Authentication and Confidentiality	7.2.2, 7.3.1.4, 5.4.3.3, 8.1, 11.3, 11.4, 8.2	O	Yes □ No □
PC2.1	WAPI Information Element (IE)	7.3.2.25	PC2:M	Yes □ No □
PC2.1.1	Multicast cipher suite	7.3.2.25	PC2.1:M	Yes □ No □
PC2.1.2	Unicast cipher suite list	7.3.2.25	PC2.1:M	Yes □ No □
PC2.1.2.1	WPI data confidentiality	8.2	PC2.1:M	Yes □ No □
PC2.1.3	Authentication key management (AKM) suite list	7.3.2.25,	PC2.1:M	Yes □ No □
PC2.1.3.1	WAI Certificate Authentication and Key Management	7.3.2.25	PC2.1:M	Yes □ No □
PC2.1.3.2	WAI Preshared Key Authentication and Key Management	7.3.2.25	PC2.1:M	Yes □ No □
PC2.1.3.3	WAI Authentication and key management	8.1	PC2.1:M	Yes □ No □
PC2.1.3.3.1	Key derivation	8.1.4.10	PC2.1:M	Yes □ No □
PC34.1.3.3.1.1	Base key derivation	8.1.4.10.1	PC2.1:M	Yes □ No □
PC2.1.3.3.1.2	Unicast key derivation	8.1.4.10.2	PC2.1:M	Yes □ No □
PC2.1.3.3.1.3	Multicast key derivation	8.1.4.10.3	PC2.1:M	Yes □ No □
PC2.1.3.3.2	Unicast Key Negotiation	8.1.4.3.2	PC2.1:M	Yes □ No □
PC2.1.3.3.3	Multicast key / STAKey announcement	8.1.4.4	PC2.1:M	Yes □ No □
PC2.1.4	WAPI capabilities	7.3.2.25,	PC2.1:M	Yes □ No □
PC2.1.5	WAPI preauthentication	8.1.4.6	PC2.1:O	Yes □ No □

Item	Protocol capability	References	Status	Support
PC2.1.6	WAPI security association management	8.1.2	PC2.1:M	Yes □ No □
PC2.1.7	BKSA caching	8.1.4.7	PC2.1:M	Yes □ No □
PC2.1.8	WAPI extended service set (ESS)	8.1.4.6	(PC2.1 and CF1):M	Yes □ No □
PC2.1.8.1	WAPI STAKey	8.1.4.5	PC2.1.8:O	Yes □ No □
PC2.1.9	WAPI independent basic service set (IBSS)	8.1.2.1.2	(PC2.1 and CF2):O	Yes □ No □
PC2.1.10	Rekeying	8.1.4.8	PC2.1.8:O	Yes □ No □
PC2.1.11	X.509 v3 certificate	8.3.1.1	PC2.1:M	Yes □ No □
PC2.1.12	GBW certificate	8.3.1.2	PC2.1:O	Yes □ No □
PC3	Distributed coordination function (DCF)	9.1, 9.2, Annex C	M	Yes □ No □
PC3.1	Network allocation vector (NAV) function	9.2.1, 9.2.5, 9.3.2.2	M	Yes □ No □
PC3.2	Interframe space usage and timing	9.2.3, 9.2.5, 9.2.10	M	Yes □ No □
PC3.3	Random Backoff function	9.2.4	M	Yes □ No □
PC3.4	DCF Access procedure	9.2.5.1, 9.2.5.5	M	Yes □ No □
PC3.5	Random Backoff procedure	9.2.5.2	M	Yes □ No □
PC3.6	Recovery procedures and retransmit limits	9.2.5.3	M	Yes □ No □
PC3.7	Request to send (RTS)/clear to send (CTS) procedure	9.2.5.4, 9.2.5.6, 9.2.5.7	M	Yes □ No □
PC3.8	Directed MAC protocol data unit (MPDU) transfer	9.2.6	M	Yes □ No □
PC3.9	Broadcast and multicast MPDU transfer	9.2.7	M	Yes □ No □
PC3.10	MAC level acknowledgment	9.2.2, 9.2.8	M	Yes □ No □

Item	Protocol capability	References	Status	Support
PC3.11	Duplicate detection and recovery	9.2.9	M	Yes □ No □

A.4.4.2 MAC frames

See annex A.4.4.2 of ISO/IEC 8802-11:2006.

A.4.4.3 Frame exchange sequences

See annex A.4.4.3 of ISO/IEC 8802-11:2006.

A.4.4.4 MAC addressing functions

See annex A.4.4.4 of ISO/IEC 8802-11:2006.

A.4.5 Frequency hopping (FH) PHY functions

See annex A.4.5 of ISO/IEC 8802-11:2006.

A.4.6 Direct sequence PHY functions

See annex A.4.6 of ISO/IEC 8802-11:2006.

A.4.7 IR baseband PHY functions

See annex A.4.7 of ISO/IEC 8802-11:2006.

A.4.8 OFDM PHY functions

See annex A.4.8 of ISO/IEC 8802-11:2006.

A.4.9 High Rate, direct sequence PHY functions

See annex A.4.9 of ISO/IEC 8802-11:2006.

A.4.10 Regulatory Domain Extensions

See annex A.4.10 of ISO/IEC 8802-11:2006.

Annex B

(informative)

Hopping sequences

See Annex B of ISO/IEC 8802-11:2006.

Annex C

(normative)

Formal description of MAC operation

See Annex C of ISO/IEC 8802-11:2006.

Annex D

(normative)

ASN.1 encoding of the MAC and PHY MIB

```
-- ****************************************
-- * IEEE 802.11 MIB
-- ****************************************
See Annex D of ISO/IEC 8802-11:2006.

-- ****************************************
-- *    GB15629dot11-WAPI-MIB
-- ****************************************
GB15629dot11-WAPI-MIB DEFINITIONS ::= BEGIN
  IMPORTS
    MODULE-IDENTITY，OBJECT-TYPE，Counter32，
    Unsigned32                                    FROM SNMPv2-SMI
    MacAddress，TruthValue，DisplayString         FROM SNMPv2-TC
    MODULE-COMPLIANCE，OBJECT-GROUP               FROM SNMPv2-CONF
    ifIndex                                       FROM RFC1213-MIB；

-- ****************************************
-- *    MODULE IDENTITY
-- ****************************************
gb15629dot11wapiMIB MODULE-IDENTITY
  ORGANIZATION
    “ChinaBWIPS（China Broadband Wireless IP Standard Group )”
  CONTACT-INFO
    “Address：P.O.BOX 88,West High-tech Development Center, Xi’an, China.
    Postcode：710075
    Tel: +86 29 8838 6220
    Fax: +86 29 8838 6218
    E-mail: bwips@chinabwips.org”
  DESCRIPTION
        “The MIB module for WAPI entities.”
  ::= { iso(1)  member-body(2)  cn(156)  bwips(11235)  GB15629(15629)  GB15629-11(11)
GB15629-11-mibs(1) 1 }

-- ************************************
-- *    Major sections
-- ************************************
wapiMIBObjects OBJECT IDENTIFIER          ::= { gb15629dot11wapiMIB 1 }
wapiMIBConformance OBJECT IDENTIFIER      ::= { gb15629dot11wapiMIB 2 }
```

```
-- ****************************************
-- *     wapiMIBObjects table
-- ****************************************
-- gb15629dot11wapiConfig                          ::= { wapiMIBObjects 1 }
-- gb15629dot11wapiConfigUnicastCiphers            ::= { wapiMIBObjects 2 }
-- gb15629dot11wapiConfigAuthenticationSuites      ::= { wapiMIBObjects 3 }
-- gb15629dot11wapiStats                           ::= { wapiMIBObjects 4 }

--**************************************************************************
-- *     gb15629dot11wapiConfig table
-- **************************************************************************
      gb15629dot11wapiConfig OBJECT-TYPE
            SYNTAX SEQUENCE OF gb15629dot11wapiConfigEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
              "Station Configuration attributes. In tablular form to
        allow for multiple instances on an agent."
        ::={ wapiMIBObjects 1 }

      gb15629dot11wapiConfigEntry OBJECT-TYPE
            SYNTAX gb15629dot11wapiConfigEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
               "An entry in the gb15629dot11wapiConfigTable. It is possible for there to be multiple
               WAPI interfaces on one agent, each with its unique MAC address. The relationship
               between a WAPI interface and an interface in the context of the Internet-standard MIB is
               one-to-one. As such, the value of an ifIndex object instance can be directly used to identify
               corresponding instances of the objects defined herein.
                 ifIndex - Each WAPI interface is represented by an ifEntry. Interface tables in this MIB
               module are indexed by ifIndex."
            INDEX {ifIndex}
        ::={ gb15629dot11wapiConfig 1 }

      gb15629dot11wapiConfigEntry::=
          SEQUENCE {
          gb15629dot11wapiConfigVersion                    Integer32，
          gb15629dot11wapiControlledAuthControl            TruthValue，
          gb15629dot11wapiControlledPortControl            INTEGER，
          gb15629dot11wapiOptionalImplemented              TruthValue，
          gb15629dot11wapiPreauthenticationImplemented     TruthValue，
          gb15629dot11wapiEnabled                          TruthValue，
```

gb15629dot11wapiPreauthenticationEnabled TruthValue，
gb15629dot11wapiConfigUnicastKeysSupported Unsigned32，
gb15629dot11wapiConfigUnicastRekeyMethod INTEGER，
gb15629dot11wapiConfigUnicastRekeyTime Unsigned32，
gb15629dot11wapiConfigUnicastRekeyPackets Unsigned32，
gb15629dot11wapiConfigMulticastCipher OCTET STRING，
gb15629dot11wapiConfigMulticastRekeyMethod INTEGER，
gb15629dot11wapiConfigMulticastRekeyTime Unsigned32，
gb15629dot11wapiConfigMulticastRekeyPackets Unsigned32，
gb15629dot11wapiConfigMulticastRekeyStrict TruthValue，
gb15629dot11wapiConfigPSKValue OCTET STRING，
gb15629dot11wapiConfigPSKPassPhrase DisplayString，
gb15629dot11wapiConfigCertificateUpdateCount Unsigned32，
gb15629dot11wapiConfigMulticastUpdateCount Unsigned32，
gb15629dot11wapiConfigUnicastUpdateCount Unsigned32，
gb15629dot11wapiConfigMulticastCipherSize Unsigned32，
gb15629dot11wapiConfigBKLifetime Unsigned32，
gb15629dot11wapiConfigBKReauthThreshold Unsigned32，
gb15629dot11wapiConfigSATimeout Unsigned32，
gb15629dot11wapiAuthenticationSuiteSelected OCTET STRING，
gb15629dot11wapiUnicastCipherSelected OCTET STRING，
gb15629dot11wapiMulticastCipherSelected OCTET STRING，
gb15629dot11wapiBKIDUsed OCTET STRING，
gb15629dot11wapiAuthenticationSuiteRequested OCTET STRING，
gb15629dot11wapiUnicastCipherRequested OCTET STRING，
gb15629dot11wapiMulticastCipherRequested OCTET STRING }

gb15629dot11wapiConfigVersion OBJECT-TYPE
SYNTAX Integer32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
“This object indicates the highest version that this entity supports.”
::= { gb15629dot11wapiConfigEntry 1}

gb15629dot11wapiControlledAuthControl OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
“This attribute indicates whether the entity enable the authentication. If the value is “0” which means “authentication disabled”, the state of the controlled port shall be set to “authenticated”; else the value is “1” which means “authentication enabled”, the state of the controlled port shall be based on the gb15629dot11wapiControlledPortControl.”

::= { gb15629dot11wapiConfigEntry 2}

gb15629dot11wapiControlledPortControl OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This attribute, when the gb15629dot11wapiControlledAuthControl is set to true, shall be effective and indicate the methods by which the port is controlled. If the value is "0" which means "auto", the state of the controlled port shall be based on the result of the authentication; else the value is "1" which means "force unauthenticated", the state of the controlled port shall be unconditionally set to "unauthenticated"."
::= { gb15629dot11wapiConfigEntry 3}

gb15629dot11wapiOptionImplemented OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This attribute, when true, shall indicate that the WAPI option is implemented; else, this attribute shall be false."
::= { gb15629dot11wapiConfigEntry 4 }

gb15629dot11wapiPreauthenticationImplemented OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This variable indicates whether the entity supports WAPI pre-authentication. This cannot be TRUE unless gb15629dot11wapiOptionImplemented is TRUE."
::= { gb15629dot11wapiConfigEntry 5 }

gb15629dot11wapiEnabled OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
" When this object is set to TRUE, this shall indicate that WAPI is enabled on this entity. The entity will advertise the WAPI Information Element in its Beacon and Probe Response frames. "
::= { gb15629dot11wapiConfigEntry 6 }

gb15629dot11wapiPreauthenticationEnabled OBJECT-TYPE
SYNTAX TruthValue

MAX-ACCESS read-write
STATUS current
DESCRIPTION
"When this object is set to TRUE, this shall indicate that WAPI pre-authentication is enabled on this entity. This object requires that dot11WAPI Enabled also be set to TRUE."
::= { gb15629dot11wapiConfigEntry 7 }

gb15629dot11wapiConfigUnicastKeysSupported OBJECT-TYPE
SYNTAX Unsigned32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object indicates the number of the unicast keys that a WAPI entity supports."
::= { gb15629dot11wapiConfigEntry 8 }

gb15629dot11wapiConfigUnicastRekeyMethod OBJECT-TYPE
SYNTAX INTEGER { disabled(1)， timeBased(2)，
packetBased(3)， timepacket-Based(4) }
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This object selects a mechanism for rekeying the WAPI USK. The default is time-based, once per day. Rekeying the USK is only applicable to an entity acting in the AE or ASUE role."
DEFVAL { timeBased }
::= { gb15629dot11wapiConfigEntry 9 }

gb15629dot11wapiConfigUnicastRekeyTime OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "seconds"
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The time in seconds after which the WAPI USK shall be refreshed. The timer shall start at the moment the USK was set using the MLME-SETWPIKEYS.request primitive."
DEFVAL { 86400 }
::= { gb15629dot11wapiConfigEntry 10 }

gb15629dot11wapiConfigUnicastRekeyPackets OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "1000 packets"
MAX-ACCESS read-write
STATUS current

DESCRIPTION
"A packet count (in 1000s of packets) after which the WAPI USK shall be refreshed. The packet counter shall start at the moment the USK was set using the MLME-SETWPIKEYS.request primitive and it shall count all packets encrypted using the current USK."
::= { gb15629dot11wapiConfigEntry 11 }

gb15629dot11wapiConfigMulticastCipher OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This object indicates the multicast cipher suite that this entity must adopt. The WAPI information element shall adopt the value of this variable, which contains a 3-octet OUI and a one-octet cipher suite identifier."
::= { gb15629dot11wapiConfigEntry 12 }

gb15629dot11wapiConfigMulticastRekeyMethod OBJECT-TYPE
SYNTAX INTEGER { disabled(1)， timeBased(2)，
packetBased(3)， timepacket-Based(4) }
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"This object selects a mechanism for rekeying the WAPI MSK. The default is time-based, once per day. Rekeying the MSK is only applicable to an entity acting in the Authenticator role."
DEFVAL { timeBased }
::= { gb15629dot11wapiConfigEntry 13 }

gb15629dot11wapiConfigMulticastRekeyTime OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "seconds"
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The time in seconds after which the WAPI MSK shall be refreshed. The timer shall start at the moment the MSK was set using the MLME-SETWPIKEYS.request primitive."
DEFVAL { 86400 }
::= { gb15629dot11wapiConfigEntry 14 }

gb15629dot11wapiConfigMulticastRekeyPackets OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "1000 packets"
MAX-ACCESS read-write

```
        STATUS current
        DESCRIPTION
            "A packet count (in 1000s of packets) after which the WAPI MSK shall be refreshed. The
            packet counter shall start at the moment the MSK was set using the
            MLME-SETWPIKEYS.request primitive and it shall count all packets encrypted using
            the current MSK."
    ::= { gb15629dot11wapiConfigEntry 15}

gb15629dot11wapiConfigMulticastRekeyStrict OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "This object signals that the MSK shall be refreshed whenever a STA leaves the BSS that
            possesses the MSK."
    ::= { gb15629dot11wapiConfigEntry 16 }

gb15629dot11wapiConfigPSKValue OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(32))
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The PSK for when WAPI in PSK mode is the selected AKM suite. In that case, the BK
            will obtain its value from this object. This object is logically write-only. Reading this
            variable shall return unsuccessful status or null or zero."
    ::= { gb15629dot11wapiConfigEntry 17 }

gb15629dot11wapiConfigPSKPassPhrase OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The PSK, for when WAPI in PSK mode is the selected AKM suite, is configured by
            gb15629dot11wapiConfigPSKValue. An alternative manner of setting the PSK uses the
            password-to-key algorithm. This variable provides a means to enter a pass-phrase. When
            this object is written, the WAPI entity shall use the password-to-key algorithm to derive a
            pre-shared and populate gb15629dot11wapiConfigPSKValue with this key. This object is
            logically write-only. Reading this variable shall return unsuccessful status or null or zero."
    ::= { gb15629dot11wapiConfigEntry 18 }

gb15629dot11wapiConfigCertificateUpdateCount OBJECT-TYPE
        SYNTAX Unsigned32 (1..4294967295)
        MAX-ACCESS read-write
        STATUS current
```

DESCRIPTION
"The number of times messages in the WAPI Certificate Authentication Handshake protocol will be retried per Certificate Authentication Handshake attempt."
DEFVAL { 3 }
::= { gb15629dot11wapiConfigEntry 19 }

gb15629dot11wapiConfigMulticastUpdateCount OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The number of times Message 1 in the WAPI Multicast Key Announcement Handshake will be retried per MSK Handshake attempt."
DEFVAL { 3 }
::= { gb15629dot11wapiConfigEntry 20 }

gb15629dot11wapiConfigUnicastUpdateCount OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The number of times Message 1 and Message 3 in the WAPI Unicast Key Negotiation Handshake will be retried per USK Handshake attempt."
DEFVAL { 3 }
::= { gb15629dot11wapiConfigEntry 21 }

gb15629dot11wapiConfigMulticastCipherSize OBJECT-TYPE
SYNTAX Unsigned32 (0..4294967295)
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This object indicates the length in bit of the multicast cipher key. This should be 256 for in SMS4. The former 128bits is the encryption key and the latter 128bits is the message integrity check key."
::= { gb15629dot11wapiConfigEntry 22 }

gb15629dot11wapiConfigBKLifetime OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS "seconds"
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The maximum lifetime of a BK in the BK cache."
DEFVAL { 43200 }

::= { gb15629dot11wapiConfigEntry 23 }

gb15629dot11wapiConfigBKReauthThreshold OBJECT-TYPE
SYNTAX Unsigned32 (1..100)
UNITS “percentage”
MAX-ACCESS read-write
STATUS current
DESCRIPTION
“The percentage of the BK lifetime that should expire before a WAI re-authentication occurs.”
DEFVAL { 70 }
::= { gb15629dot11wapiConfigEntry 24 }

gb15629dot11wapiConfigSATimeout OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
UNITS “seconds”
MAX-ACCESS read-write
STATUS current
DESCRIPTION
“The maximum time a security association shall take to set up.”
DEFVAL { 60 }
::= { gb15629dot11wapiConfigEntry 25 }

gb15629dot11wapiAuthenticationSuiteSelected OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
“The selector of the last AKM suite negotiated.”
::= { gb15629dot11wapiConfigEntry 26 }

gb15629dot11wapiUnicastCipherSelected OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
“The selector of the last unicast cipher suite negotiated.”
::= { gb15629dot11wapiConfigEntry 27 }

gb15629dot11wapiMulticastCipherSelected OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-only
STATUS current
DESCRIPTION

```
                "The selector of the last multicast cipher suite negotiated"
    ::= { gb15629dot11wapiConfigEntry 28 }

  gb15629dot11wapiBKIDUsed OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(16))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The selector of the last BKID used in the last Unicast Key Negotiation Handshake."
    ::= { gb15629dot11wapiConfigEntry 29 }

  gb15629dot11wapiAuthenticationSuiteRequested OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The selector of the last AKM suite requested."
    ::= { gb15629dot11wapiConfigEntry 30 }

  gb15629dot11wapiUnicastCipherRequested OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The selector of the last unicast cipher suite requested"
    ::= { gb15629dot11wapiConfigEntry 31 }

  gb15629dot11wapiGroupCipherRequested OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "The selector of the last multicast cipher suite requested."
    ::= { gb15629dot11wapiConfigEntry 32 }
--*********************************************************************
-- *    End of    gb15629dot11wapiConfig table
-- *********************************************************************


-- *********************************************************************
-- *    gb15629dot11wapiConfigUnicastCiphers table
-- *********************************************************************
  gb15629dot11wapiConfigUnicastCiphers OBJECT-TYPE
        SYNTAX SEQUENCE OF gb15629dot11wapiConfigUnicastCiphersEntry
```

MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"This table lists the unicast cipher suites supported by this entity. It allows enabling and disabling of each unicast cipher suite by network management. The unicast cipher suite list in the WAPI Information Element is formed using the information in this table."
::= { wapiMIBObjects 2 }

gb15629dot11wapiConfigUnicastCiphersEntry OBJECT-TYPE
SYNTAX gb15629dot11wapiConfigUnicastCiphersEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The table entry, indexed by the interface index (or all interfaces) and the unicast cipher suite."
INDEX { gb15629dot11wapiConfigIndex， gb15629dot11wapiConfigUnicastCipherIndex }
::= { gb15629dot11wapiConfigUnicastCiphersTable 1 }

gb15629dot11wapiConfigUnicastCiphersEntry ::=
SEQUENCE {
gb15629dot11wapiConfigUnicastCipherIndex Unsigned32，
gb15629dot11wapiConfigUnicastCipher OCTET STRING，
gb15629dot11wapiConfigUnicastCipherEnabled TruthValue，
gb15629dot11wapiConfigUnicastCipherSize Unsigned32 }

gb15629dot11wapiConfigUnicastCipherIndex OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
" The auxiliary index into the gb15629dot11wapiConfigUnicastCiphersTable."
::= { gb15629dot11wapiConfigUnicastCiphersEntry 1 }

gb15629dot11wapiConfigUnicastCipher OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The selector of a supported unicast cipher suite. It consists of an OUI (the first 3 octets) and a cipher suite identifier (the last octet)."
::= { gb15629dot11wapiConfigUnicastCiphersEntry 2 }

gb15629dot11wapiConfigUnicastCipherEnabled OBJECT-TYPE
SYNTAX TruthValue

```
            MAX-ACCESS read-write
            STATUS current
            DESCRIPTION
                "This object enables or disables the unicast cipher."
        ::= { gb15629dot11wapiConfigUnicastCiphersEntry 3 }

    gb15629dot11wapiConfigUnicastCipherSize OBJECT-TYPE
            SYNTAX Unsigned32 (0..4294967295)
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "This object indicates the length in bit of the unicast cipher key. This should be 256 for
                SMS4. The former 128bits is the encryption key and the latter 128bits is the message
                integrity check key."
        ::= { gb15629dot11wapiConfigUnicastCiphersEntry 4 }
-- ****************************************************************
-- *    End of gb15629dot11wapiConfigUnicastCiphers
-- ****************************************************************


-- ****************************************************************
-- *    gb15629dot11wapiConfigAuthenticationSuites table
-- ****************************************************************
    gb15629dot11wapiConfigAuthenticationSuites OBJECT-TYPE
            SYNTAX SEQUENCE OF gb15629dot11wapiConfigAuthenticationSuitesEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "This table lists the AKM suites supported by this entity. Each AKM suite can be
                individually enabled and disabled. The AKM suite list in the WAPI information element is
                formed using the information in this table."
        ::= { wapiMIBObjects 3 }

    gb15629dot11wapiConfigAuthenticationSuitesEntry OBJECT-TYPE
            SYNTAX gb15629dot11wapiConfigAuthenticationSuitesEntry
            MAX-ACCESS not-accessible
            STATUS current
            DESCRIPTION
                "An entry in the gb15629dot11WAPIConfigAuthenticationSuitesTable."
            INDEX { gb15629dot11wapiConfigAuthenticationSuiteIndex }
        ::= { gb15629dot11wapiConfigAuthenticationSuitesTable 1 }

    gb15629dot11wapiConfigAuthenticationSuitesEntry ::=
            SEQUENCE {
```

```
        gb15629dot11wapiConfigAuthenticationSuiteIndex      Unsigned32，
        gb15629dot11wapiConfigAuthenticationSuite           OCTET STRING，
        gb15629dot11wapiConfigAuthenticationSuiteEnabled    TruthValue }

    gb15629dot11wapiConfigAuthenticationSuiteIndex OBJECT-TYPE
        SYNTAX Unsigned32 (1..4294967295)
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            “The auxiliary variable used as an index into the
            gb15629dot11WAPIConfigAuthenticationSuitesTable.”
      ::= { gb15629dot11wapiConfigAuthenticationSuitesEntry 1 }

    gb15629dot11wapiConfigAuthenticationSuite OBJECT-TYPE
        SYNTAX OCTET STRING (SIZE(4))
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            “The selector of an AKM suite. It consists of an OUI (the first 3 octets) and a cipher suite
            identifier (the last octet).”
      ::= { gb15629dot11wapiConfigAuthenticationSuitesEntry 2 }

    gb15629dot11wapiConfigAuthenticationSuiteEnabled OBJECT-TYPE
        SYNTAX TruthValue
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            “This variable indicates whether the corresponding AKM suite is enabled/disabled.”
      ::= { gb15629dot11wapiConfigAuthenticationSuitesEntry 3 }
-- ************************************************************
-- *    End of gb15629dot11wapiConfigAuthenticationSuites
-- ************************************************************


-- ************************************************************
-- *    gb15629dot11wapiStats table
-- ************************************************************
    gb15629dot11wapiStats OBJECT-TYPE
        SYNTAX SEQUENCE OF gb15629dot11wapiStatsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            “This table maintains per-STA statistics in a WAPI. The entry with
        gb15629dot11wapiStatsSTAAddress set to FF-FF-FF-FF-FF-FF shall contain statistics for
```

broadcast/multicast traffic."
::= { wapiMIBObjects 4 }

gb15629dot11wapiStatsEntry OBJECT-TYPE
SYNTAX gb15629dot11wapiStatsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An entry in the gb15629dot11wapiStatsTable."
INDEX { gb15629dot11wapiConfigIndex，gb15629dot11wapiStatsIndex }
::= { gb15629dot11wapiStats 1 }

gb15629dot11wapiStatsEntry ::=
SEQUENCE {
gb15629dot11wapiStatsIndex Unsigned32，
gb15629dot11wapiStatsSTAAddress MacAddress，
gb15629dot11wapiStatsVersion Unsigned32，
gb15629dot11wapiStatsControlledPortStatus TruthValue，
gb15629dot11wapiStatsSelectedUnicastCipher OCTET STRING，
gb15629dot11wapiStatsWPIReplayCounters Counter32，
gb15629dot11wapiStatsWPIDecryptableErrors Counter32，
gb15629dot11wapiStatsWPIMICErrors Counter32，
gb15629dot11wapiStatsWAISignatureErrors Counter32，
gb15629dot11wapiStatsWAIHMACErrors Counter32，
gb15629dot11wapiStatsWAIAuthenticationResultFailures Counter32，
gb15629dot11wapiStatsWAIDiscardCounters Counter32，
gb15629dot11wapiStatsWAITimeoutCounters Counter32，
gb15629dot11wapiStatsWAIFormatErrors Counter32，
gb15629dot11wapiStatsWAICertificateHandshakeFailures Counter32，
gb15629dot11wapiStatsWAIUnicastHandshakeFailures Counter32，
gb15629dot11wapiStatsWAIMulticastHandshakeFailures Counter32}

gb15629dot11wapiStatsIndex OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An auxiliary index into the gb15629dot11wapiStatsTable."
::= { gb15629dot11wapiStatsEntry 1 }

gb15629dot11wapiStatsSTAAddress OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-only
STATUS current

DESCRIPTION
"STA's MAC address."
::= { gb15629dot11wapiStatsEntry 2 }

gb15629dot11wapiStatsVersion OBJECT-TYPE
SYNTAX Unsigned32 (1..4294967295)
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The WAPI version with which the STA associated."
::= { gb15629dot11wapiStatsEntry 3 }

gb15629dot11wapiStatsControlledPortStatus OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This attribute indicates the state of the controlled port in a authenticator subsystem. If value is "1", the state is set to"authenticated"; else, the state is set to "unauthenticated"."
::= { gb15629dot11wapiStatsEntry 4 }

gb15629dot11wapiStatsSelectedUnicastCipher OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(4))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The unicast cipher suite selector used during association."
::= { gb15629dot11wapiStatsEntry 5 }

gb15629dot11wapiStatsWPIReplayCounter OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of received WPI MPDUs discarded by the replay mechanism."
::= { gb15629dot11wapiStatsEntry 6 }

gb15629dot11wapiStatsWPIDecryptableErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of received MPDUs discarded by the WPI decryption algorithm because of no valid key."

::= { gb15629dot11wapiStatsEntry 7 }

gb15629dot11wapiStatsWPIMICErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of received MPDUs discarded because of the MIC check error during the WPI decryption."
::= { gb15629dot11wapiStatsEntry 8 }

gb15629dot11wapiStatsWAISignatureErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the signature in the received WAI packet is wrong."
::={ gb15629dot11wapiStatsEntry 9 }

gb15629dot11wapiStatsWAIHMACErrors OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the message authentication code in the received WAI packet is wrong."
::={ gb15629dot11wapiStatsEntry 10}

gb15629dot11wapiStatsWAIAuthenticationResultFailures OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the WAI authentication is unsuccessful."
::={ gb15629dot11wapiStatsEntry 11}

gb15629dot11wapiStatsWAIDiscardCounters OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This counter shall increment when the received WAI packet is discarded."
::={ gb15629dot11wapiStatsEntry 12}

```
    gb15629dot11wapiStatsWAITimeoutCounters OBJECT-TYPE
            SYNTAX    Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "This counter shall increment when the WAI packet is timeout."
        ::={ gb15629dot11wapiStatsEntry 13}

    gb15629dot11wapiStatsWAIFormatErrors    OBJECT-TYPE
            SYNTAX    Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
              "This counter shall increment when there exists format error in the WAI packet."
        ::={ gb15629dot11wapiStatsEntry 14}

    gb15629dot11wapiStatsWAICertificateHandshakeFailures    OBJECT-TYPE
            SYNTAX    Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "This counter shall increment when the WAI certificate authentication is unsuccessful."
        ::={ gb15629dot11wapiStatsEntry 15}

    gb15629dot11wapiStatsWAIUnicastHandshakeFailures    OBJECT-TYPE
            SYNTAX    Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "This counter shall increment when the WAI unicast key negotiation process is
            unsuccessful."
        ::={ gb15629dot11wapiStatsEntry 16}

    gb15629dot11wapiStatsWAIMulticastHandshakeFailures    OBJECT-TYPE
            SYNTAX    Counter32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                "This counter shall increment when the WAI multicast key announcement process is
            unsuccessful."
        ::={ gb15629dot11wapiStatsEntry 17}
-- ************************************************************************
-- *    End of gb15629dot11wapiStats
-- ************************************************************************
```

```
-- ****************************************
-- *    Conformance information -- gb15629.11 WAPI MIB
-- ****************************************
gb15629dot11wapiGroups OBJECT IDENTIFIER ::= { wapiMIBConformance 1 }
gb15629wapiCompliances OBJECT IDENTIFIER::= { wapiMIBConformance 2 }

-- ****************************************************************************
-- * Conformance information – WAPI
-- ****************************************************************************

   gb15629dot11wapiCompliance MODULE-COMPLIANCE
          STATUS current
          DESCRIPTION
               “Implement conformance statement of the SNMPv2 entity in the WAPI MIB.”
          MODULE
          MANDATORY-GROUPS {
          gb15629dot11wapiBase }
          -- OPTIONAL-GROUPS {gb15629dot11wapiBKcachingGroup }
     ::= { gb15629WapiCompliances 1 }

-- ****************************************************************************
-- * Groups - units of conformance– WAPI
-- ****************************************************************************
   gb15629dot11wapiBase OBJECT-GROUP
          OBJECTS {
               gb15629dot11wapiConfigVersion，
               gb15629dot11wapiControlledAuthControl，
               gb15629dot11wapiControlledPortControl，
               gb15629dot11wapiOptionalImplemented，
               gb15629dot11wapiPreauthenticationImplemented，
               gb15629dot11wapiEnabled，
               gb15629dot11wapiPreauthenticationEnabled，
               gb15629dot11wapiConfigUnicastKeysSupported，
               gb15629dot11wapiConfigUnicastRekeyMethod，
               gb15629dot11wapiConfigUnicastRekeyTime，
               gb15629dot11wapiConfigUnicastRekeyPackets，
               gb15629dot11wapiConfigMulticastCipher，
               gb15629dot11wapiConfigMulticastRekeyMethod，
               gb15629dot11wapiConfigMulticastRekeyTime，
               gb15629dot11wapiConfigMulticastRekeyPackets，
               gb15629dot11wapiConfigMulticastRekeyStrict，
               gb15629dot11wapiConfigPSKValue，
```

gb15629dot11wapiConfigPSKPassPhrase，
gb15629dot11wapiConfigCertificateUpdateCount，
gb15629dot11wapiConfigMulticastUpdateCount，
gb15629dot11wapiConfigUnicastUpdateCount，
gb15629dot11wapiConfigMulticastCipherSize，
gb15629dot11wapiConfigUnicastCipher，
gb15629dot11wapiConfigUnicastCipherEnabled，
gb15629dot11wapiConfigUnicastCipherSize，
gb15629dot11wapiConfigAuthenticationSuite，
gb15629dot11wapiConfigAuthenticationSuiteEnabled，
gb15629dot11wapiConfigSATimeout，
gb15629dot11wapiAuthenticationSuiteSelected，
gb15629dot11wapiUnicastCipherSelected，
gb15629dot11wapiMulticastCipherSelected，
gb15629dot11wapiBKIDUsed，
gb15629dot11wapiAuthenticationSuiteRequested，
gb15629dot11wapiUnicastCipherRequested，
gb15629dot11wapiMulticastCipherRequested，
gb15629dot11wapiStatsSTAAddress，
gb15629dot11wapiStatsVersion，
gb15629dot11wapiStatsControlledPortStatus，
gb15629dot11wapiStatsSelectedUnicastCipher，
gb15629dot11wapiStatsWPIReplayCounters，
gb15629dot11wapiStatsWPIDecryptableErrors，
gb15629dot11wapiStatsWPIMICErrors，
gb15629dot11wapiStatsWAISignatureErrors，
gb15629dot11wapiStatsWAIHMACErrors，
gb15629dot11wapiStatsWAIAuthenticationResultFailures，
gb15629dot11wapiStatsWAIDiscardCounters，
gb15629dot11wapiStatsWAITimeoutCounters，
gb15629dot11wapiStatsWAIFormatErrors，
gb15629dot11wapiStatsWAICertificateHandshakeFailures，
gb15629dot11wapiStatsWAIUnicastHandshakeFailures，
gb15629dot11wapiStatsWAIMulticastHandshakeFailures
}
STATUS current
DESCRIPTION
" The gb15629dot11wapiBase class provides the necessary support to manage the WAPI function of STA."
::= { gb15629dot11WapiGroups 28 }

gb15629dot11wapiBKcachingGroup OBJECT-GROUP
OBJECTS {gb15629dot11wapiConfigBKLifetime，
gb15629dot11wapiConfigBKReauthThreshold}

```
        STATUS current
        DESCRIPTION
            " The gb15629dot11wapiBKcachingGroup class provides the necessary support to manage
            the BK cache function of STA."
    ::= { gb15629dot11WapiGroups 29 }

--*****************************************************************************
--*    End of gb15629dot11-WAPI-MIB
--*****************************************************************************
  END
```

Annex E
(informative)

Bibliography
See Annex E of ISO/IEC 8802-11:2006.

Annex F
(informative)

High Rate PHY/FH interoperability
See Annex F of ISO/IEC 8802-11:2006.

Annex G
(informative)

An example of encoding a frame for OFDM PHY
See Annex G of ISO/IEC 8802-11:2006.

Annex H
(informative)
Reference implementations of the frame authentication algorithm and the key derivation algorithm and the test vectors

H.1 Frame authentication algorithm

H.1.1 Reference implementation

*int hmac_sha256(unsigned char *text, int text_len, byte *key, unsigned key_len, byte *digest,unsigned digest_length)*

*/**

a) unsigned char *text , indicates the text to be calculated by HMAC;

b) unsigned text_len , indicates the length of the text to be calculated by HMAC (in octet);

c) byte *key , indicates the key used in the HMAC computation;

d) unsigned key_len, indicates the length of the key used in the HMAC computation (in octet);

e) byte *digest , indicates the output digest of HMAC;

f) unsigned digest_length , specifies the length of the output digest of HMAC (in octet) which must be less than that of the hash algorithm sha256 (in octet);

g) The returned value. The non-zero values indicate the practical length of the output digest, and the value of 0 indicates a failure.

*/

{

```
byte real_key[SHA256_BLOCK_SIZE];
byte ipad[SHA256_BLOCK_SIZE];
byte opad[SHA256_BLOCK_SIZE];
byte temp_digest1[SHA256_DIGEST_SIZE];
byte temp_digest2[SHA256_DIGEST_SIZE];
CONTX input_data[2];
unsigned i;

if digest_length>SHA256_DIGEST_SIZE)
    return 0;

for(i=0;i< SHA256_BLOCK_SIZE;i++){
    real_key[i]=0;
    ipad[i]=0x36;
    opad[i]=0x5c;
}

/* if key_len is larger than hash block size, key is hashed first to make its length is equal hash block size
*/
```

```
    if(key_len> SHA256_BLOCK_SIZE){
        input_data[0].buff=key;
        input_data[0].length=key_len;
        SHA256(input_data,1,real_key);
        key_len= SHA256_ BLOCK _SIZE;
    }
    else
        memcpy(real_key,key,key_len);

    for(i=0;i< SHA256_BLOCK_SIZE;i++){
        ipad[i]^=real_key[i];
        opad[i]^=real_key[i];
    }

    /*sha256(Key xor ipad,text)=temp_digest1 */
    input_data[0].buff=ipad;
    input_data[0].length= SHA256_BLOCK_SIZE;
    input_data[1].buff=text;
    input_data[1].length=text_len;
    SHA256(input_data,2,temp_digest1);

    /*sha256(Key xor opad,temp_digest1)=temp_digest2 */
    input_data[0].buff=opad;
    input_data[0].length= SHA256_BLOCK_SIZE;
    input_data[1].buff=temp_digest1;
    input_data[1].length SHA256_DIGEST_SIZE;
    SHA256(input_data,2,temp_digest2);

    /*output the digest of required length */
    memcpy(digest,temp_digest2,digest_length);
    return digest_length;
}
```

H.1.2 Test vectors

Test vector 1	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20
Length of key	32

Data	**abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopqabcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq** 0x61 62 63 64 62 63 64 65 63 64 65 66 64 65 66 67 65 66 67 68 66 67 68 69 67 68 69 6a 68 69 6a 6b 69 6a 6b 6c 6a 6b 6c 6d 6b 6c 6d 6e 6c 6d 6e 6f 6d 6e 6f 70 6e 6f 70 71 61 62 63 64 62 63 64 65 63 64 65 66 64 65 66 67 65 66 67 68 66 67 68 69 67 68 69 6a 68 69 6a 6b 69 6a 6b 6c 6a 6b 6c 6d 6b 6c 6d 6e 6c 6d 6e 6f 6d 6e 6f 70 6e 6f 70 71
Length of data	112
Digest	0x47 03 05 fc 7e 40 fe 34 d3 ee b3 e7 73 d9 5a ab 73 ac f0 fd 06 04 47 a5 eb 45 95 bf 33 a9 d1 a3

Test vector 2	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
Length of key	37
Data	0xcd repeated by 50 times
Length of data	50
Digest	0xd4 63 3c 17 f6 fb 8d 74 4c 66 de e0 f8 f0 74 55 6e c4 af 55 ef 07 99 85 41 46 8e b4 9b d2 e9 17

Test vector 3	
Key	0x0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b 0b
Length of key	32
Data	**Hi There** 0x48 69 20 54 68 65 72 65
Length of data	8
Digest	0x19 8a 60 7e b4 4b fb c6 99 03 a0 f1 cf 2b bd c5 ba 0a a3 f3 d9 ae 3c 1c 7a 3b 16 96 a0 b6 8c f7

Test vector 4	
Key	0x4a 65 66 65
Length of key	4
Data	**what do ya want for nothing?** 0x77 68 61 74 20 64 6f 20 79 61 20 77 61 6e 74 20 66 6f 72 20 6e 6f 74 68 69 6e 67 3f
Length of data	28

Digest	0x5b dc c1 46 bf 60 75 4e 6a 04 24 26 08 95 75 c7 5a 00 3f 08 9d 27 39 83 9d ec 58 b9 64 ec 38 43

H.2 Key derivation algorithm

H.2.1 Reference implementation

void KD_hmac_sha256(byte *text,unsigned text_len,byte *key,unsigned key_len,byte *output,unsigned length)

/*

a) byte *text , indicates the input text of the key derivation algorithm;

b) unsigned text_len , indicates the length of the input text (in octet);

c) byte *key, indicates the input key of the key derivation algorithm;

d) unsigned key_len , indicates the length of the input key (in octet);

e) byte *output , indicates the output of the key derivation algorithm;

f) unsigned length , indicates the length of the output of the key derivation algorithm (in octet).

***/**

{

int i;

for(i=0;length/SHA256_DIGEST_SIZE;i++,length-=SHA256_DIGEST_SIZE){

hmac(MHASH_SHA256,text,text_len,key,key_len,&output[i*SHA256_DIGEST_SIZE],SHA256_DIGEST_SIZE);

text=&output[i*SHA256_DIGEST_SIZE];

text_len=SHA256_DIGEST_SIZE;

}

if(length>0)

hmac(MHASH_SHA256,text,text_len,key,key_len,&output[i*SHA256_DIGEST_SIZE],length);

}

H.2.2 Test vectors

Test vector 1	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20
Length of key	32
Data	**pairwise key expansion for infrastructure unicast** 0x70 61 69 72 77 69 73 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 69 6e 66 72 61 73 74 72 75 63 74 75 72 65 20 75 6e 69 63 61 73 74
Length of data	49
Output	0xe3 a6 45 46 f2 d1 f5 ee b7 d1 ee 06 d2 c9 e5 4a 2c c9 d6 ce c3 b7 6f fd 62 63 f4 26 dc 25 39 af bd 98 80 a5 27 a1 b5 85 59 4b 57 ce 33 21 4f 0c
Length of output	48

Test vector 2	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
Length of key	37
Data	**pairwise key expansion for infrastructure unicast** 0x70 61 69 72 77 69 73 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 69 6e 66 72 61 73 74 72 75 63 74 75 72 65 20 75 6e 69 63 61 73 74
Length of data	49
Output	0x3b 6e ca 4f 08 76 c4 3a b3 1b 26 3f 2c 38 b8 81 21 b5 68 e5 f8 fd 1d 4c fa 4c 7f 8c 60 97 04 3d 7b 40 a8 63 b9 43 b9 f5 bb 37 2f 3a dd a5 da 27
Length of output	48

Test vector 3	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15
Length of key	16
Data	**pairwise key expansion for infrastructure unicast** 0x70 61 69 72 77 69 73 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 69 6e 66 72 61 73 74 72 75 63 74 75 72 65 20 75 6e 69 63 61 73 74
Length of data	49
Output	0xbc 29 f3 e6 09 1f 6a c9 0b a0 20 61 92 12 48 69 5f ee ff 1a 4c ab 53 3b 11 67 d8 54 5f 93 5f 28 11 84 c9 bb 32 f9 87 b9 86 81 0f fb 17 c4 10 f5
Length of output	48

Test vector 4	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20
Length of key	32
Data	**group key expansion for multicast and broadcast** 0x67 72 6f 75 70 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 6d 75 6c 74 69 63 61 73 74 20 61 6e 64 20 62 72 6f 61 64 63 61 73 74
Length of data	47
Output	0x20 8f 72 54 a4 bf 56 f0 fa 49 5f e1 0c 99 15 05 92 ed 79 df 57 74 a9 6e 13 97 1e c4 a1 5e 16 a7 ed 75 f5 e5 44 bb d3 35 67 eb 88 e7 83 24 a9 d2

Length of output	48
Test vector 5	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
Length of key	37
Data	**group key expansion for multicast and broadcast** 0x67 72 6f 75 70 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 6d 75 6c 74 69 63 61 73 74 20 61 6e 64 20 62 72 6f 61 64 63 61 73 74
Length of data	47
Output	0x33 32 61 7a 90 8e a5 a0 7f fa 1d 23 79 f3 d8 3e 8b e9 14 1f 15 53 8f d3 ef de 58 01 19 e8 c5 09 5d 25 b2 d3 0a c7 a6 35 ad b4 3c 6c ac f0 aa 2b
Length of output	48

Test vector 6	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15
Length of key	16
Data	**group key expansion for multicast and broadcast** 0x67 72 6f 75 70 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 6d 75 6c 74 69 63 61 73 74 20 61 6e 64 20 62 72 6f 61 64 63 61 73 74
Length of data	47
Output	0xf2 cb f1 1c 6d 40 b8 09 d0 c0 ed 48 2a 4a 1b 6a 15 1a f1 fb 4c 80 f9 80 5c 93 e5 6e b1 cf 5c b5 ec c1 3e 7a bc af e0 a7 d2 59 5d 51 9b 76 9a 24
Length of output	48

Test vector 7	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20
Length of key	32
Data	**pre-share key expansion for ad-hoc network** 0x70 72 65 2d 73 68 61 72 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 61 64 68 6f 63 20 6e 65 74 77 6f 72 6b
Length of data	41
Output	0xc0 7a d8 32 25 2a 0c 14 76 18 f4 c0 d0 6b 35 f4 f6 d6 73 5d 1a a3 8e 47 9a 7e e0 ac 1c 0c 38 5b 2d 33 28 74 1e 4d a0 c8 76 fc 6c c9 e3 60 c8 d7

Length of output	48

Test vector 8	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25
Length of key	37
Data	**pre-share key expansion for adhoc network** 0x70 72 65 2d 73 68 61 72 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 61 64 68 6f 63 20 6e 65 74 77 6f 72 6b
Length of data	41
Output	0x f0 0b ee f2 f5 5f 85 d8 ee b0 6f 8c c4 1b e6 0e c2 69 f5 82 9a 0b 6e fb 2d 9b 49 5e b1 87 d3 58 59 68 88 c3 d2 6f 94 9f 8d 2e 41 fe bc bb b9 9a
Length of output	48

Test vector 9	
Key	0x01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15
Length of key	16
Data	**pre-share key expansion for adhoc network** 0x70 72 65 2d 73 68 61 72 65 20 6b 65 79 20 65 78 70 61 6e 73 69 6f 6e 20 66 6f 72 20 61 64 68 6f 63 20 6e 65 74 77 6f 72 6b
Length of data	41
Output	0x05 8e b8 7c ff 82 66 47 de 50 7b 14 17 ac 99 6e b5 7f cf 11 fd fc 83 be 59 d5 85 f4 a7 3e 69 7d d4 38 e3 34 fe bb 06 7d 14 6f 01 31 a6 96 4f 26
Length of output	48

Annex I
(Informative)
The example of WAI parameters and WPI block cryptographic algorithm

I.1 Principle

Cryptographic algorithms and ECC parameter to be applied in information security mechanism may be subject to national and regional regulations. It shall conform to national laws and regulations, and can be chosen according to specific requirements in different countries and regions. An appropriate block algorithm shall be adopted in WPI, and which type of algorithm to be adopted is optional. The appropriate ECC parameters shall be adopted in WAI, and which parameter to be adopted is optional.

An appropriate algorithm means that this algorithm should meet the requirements about the block size and the key length, i.e. 128 bits.

I.2 Algorithm used in China

In this standard, no specific cryptographic algorithms are defined. Cryptographic algorithms to be applied in WPI may be subject to national and regional regulations, and they should have a 128-bit block size and a 128-bit key length.
In China, the adopted cryptographic algorithm is SMS4, which has a 128-bit block size and a 128-bit key length.
The corresponding OUI of SMS4 is 00-14-72 and the Suite Type is 1. The detailed information of SMS4 can be obtained from the website http://www.oscca.gov.cn.

I.3 ECC parameters used in China

In this standard, no specific ECC parameters are defined. ECC parameters to be applied in WAI may be subject to national and regional regulations.
The OID corresponding to ECC parameters chosen in China is 1.2.156.1001.5.40.1. The OID 1.2.156.1001.5.40.1 is also applied in GBW and ECDH. The detailed information can be obtained from the website http://www.oscca.gov.cn.