802.11bi Draft Specification

|  |
| --- |
| Proposed spec texts for key creation using authentication frame for 802.1X |
| Date: 2024-04-30 |
| Author(s): |
| Name | Affiliation | Address | Phone | Email |
| Po-Kai Huang | Intel |  |  | po-kai.huang@intel.com |
| Ido Ouzieli |  |  |  |  |
| Johannes Berg |  |  |  |  |
| Ilan Peer |  |  |  |  |
| Robert Stacey |  |  |  |  |

Abstract

This submission proposes spec text based on the following passed requirement.

* ***11bi shall define a mechanism for a CPE Client and CPE AP to establish keys from an Authentication exchange which can then be used to protect the (Re)Association Request/Response.***

Revision History:

* Rev 0: Initial version of the document
* Rev 1: Revision to add missing period in the bullet as suggested in the teleconference call.
* Rev 2: Revision to accommodate the comments from Jouni. Remove “as defined in IETF RFC 8110”. Other changes are marked with green.

***Editing instructions formatted like this are intended to be copied into the TGbi Draft (i.e. they are instructions to the 802.11 editor on how to merge the text with the baseline documents).***

***TGbi Editor: Editing instructions preceded by “TGbi Editor” are instructions to the TGbi editor to modify or insert material in the TGbi draft. As a result of adopting the changes, the TGbi editor will execute the instructions rather than copy them to the TGbi Draft.***

**Discussion:**

We utilize the authentication frame exchange for 802.1X using authentication frame to describe how PTKSA can be created by reusing authentication frame exchange of 802.1X using authentication frame to encrypt the coming reassociation request/response frame.

* If both sides support encryption of (re)association request/response frame, then PTKSA can be derived after 802.1X authentication frame exchange using authentication frame.
* To incorporate PFS mechanisms introduced in PASN to the key derivation, we simply add Diffie-Hellman Parameter element in authentication frame 1 and authentication frame 2 to derive DHss and have it as part of the key derivation for PTKSA.
* The first authentication frame will carry Diffie-Hellman Parameter element from the supplicant. After receiving the parameters from the supplicant, authenticator can then derive DHss with its own parameters and be ready to derive PTKSA when EAP Success is sent.
* The first authentication frame will also carry RSNE to indicate pairwise cipher to be used to encrypt the (re)association request/response frame and carry Nonce element to indicate SNonce to be used during the PTKSA derivation.
* The first authentication frame will carry RSNXE to indicate things like WUR is supported, so KDK can be derived during PTKSA if both sides support WUR.
* The second authentication frame will carry Diffie-Hellman Parameter element from the authenticator. After receiving the parameters from the authenticator, supplicant can then derive DHss and be ready to derive PTKSA when EAP Success is received.
* The second authentication frame will also carry Nonce element to indicate ANonce to be used during the PTKSA derivation.
* As a result, (re)association request/response frame can be encrypted.



We note below that based on RFC 3748, EAP success or EAP failure is always sent by the authenticator at the end of EAP protocol to indicate success or failure of the authentication exchange.

<https://datatracker.ietf.org/doc/html/rfc3748>

***2. Extensible Authentication Protocol (EAP)***

*The EAP authentication exchange proceeds as follows:
…..*

* *[4] The conversation continues until the authenticator cannot authenticate the peer (unacceptable Responses to one or more Requests), in which case the authenticator implementation MUST transmit an EAP Failure (Code 4). Alternatively, the authentication conversation can continue until the authenticator determines that successful authentication has occurred, in which case the authenticator MUST transmit an EAP Success (Code 3).*

**Proposed Texts:**

**TGbi Editor: *Instruction Change clause 9.4.2.188 as below and change “FILS Nonce” to “Nonce” in all instances throughout the document:***

* Nonce element

The Nonce element is used for exchanging an additional source of randomness in the authentication exchange. The format of the Nonce element is shown in Figure 9-757 (Nonce element format).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Element ID | Length | Element ID Extension | Nonce |
| Octets: | 1 | 1 | 1 | 16 |
| * Nonce element format
 |

The Element ID, Length, and Element ID Extension fields are defined in 9.4.2.1 (General).

The Nonce field contains randomly generated data.

* Authentication frame format

**TGbi Editor: *Instruction:Modify Table 9-71 as follows:***

***Insert new rows at the end of Table 9-71 (Presence of fields and elements in Authentication frames) as follows (not all lines shown):***

***Table 9-71 - Presence of fields and elements in Authentication frames***

|  |  |  |  |
| --- | --- | --- | --- |
| **Authentication algorithm** | **Authentication transaction sequence number** | **Status code** | **Presence of fields and elements**  |
| IEEE 802.1X authentication | 1 | Reserved  | The Length of Encapsulation field is present.The Encapsulation field is present only when the Length of Encapsulation field indicates a non-zero value.The AKM Suite Selector element is present as defined in 12.14.4 (IEEE 802.1X authentication utilizing Authentication frames).The RSNE is present as defined in 12.14.7.2 (802.1X).The RSNXE is present as defined in 12.14.7.2 (802.1X).The Nonce element is present as defined in 12.14.7.2 (802.1X).The Diffie-Hellman Parameter element is present as defined in 12.14.7.2 (802.1X). |
| IEEE 802.1X authentication(#Ed) | 2 | SUCCESS | The Length of Encapsulation field is present.The Encapsulation field is present only when the Length of Encapsulation field indicates a non-zero value.The AKM Suite Selector element is present as defined in 12.14.4 (IEEE 802.1X authentication utilizing Authentication frames).The RSNE is present as defined in 12.14.7.2 (802.1X).The Nonce element is present as defined in 12.14.7.2 (802.1X).The Diffie-Hellman Parameter element is present as defined in 12.14.7.2 (802.1X). |
| IEEE 802.1X authentication(#Ed) | 2 | Not SUCCESS(#Ed) | The Length of Encapsulation field is present.The Encapsulation field is present only when the Length of Encapsulation field indicates a non-zero value. |
| IEEE 802.1X authentication(#Ed) | 3 | SUCCESS | The Length of Encapsulation field is present.The Encapsulation field is present only when the Length of Encapsulation field indicates a non-zero value. |
| IEEE 802.1X authentication(#Ed) | 3 | Not SUCCESS(#Ed) | The Length of Encapsulation field is present.The Encapsulation field is present only when the Length of Encapsulation field indicates a non-zero value. |
| IEEE 802.1X authentication | > 3 | Status | The Length of Encapsulation field is present.The Encapsulation field is present only when the Length of Encapsulation field indicates a non-zero value. |

**12.14 Client Privacy Enhancement**

**TGbi Editor: *Instruction: Insert 12.14.7. 802.1X as shown below***

**12.14.7 Key derivation with Authentication frame exchange**

**12.14.7.2 802.1X**

If an authentication originator or an authentication responder defined in 12.14.4 (IEEE 802.1X authentication utilizing Authentication frames)) sets the Encryption of the Frame Body Field of the (Re)Association Request/Response Frame Support subfield in the RSNXE to 1, then the authentication originator or the authentication responder supports the additional rules defined in this subclause when performing 802.1X Authentication frame exchange.

An authentication originator that sets the Encryption of the Frame Body Field of the (Re)Association Request/Response Frame Support subfield in the RSNXE to 1, has the corresponding SME to act as the Supplicant, sees the Encryption of the Frame Body Field of the (Re)Association Request/Response Frame Support subfield in the RSNXE of the authentication responder set to 1, and intends to continue association after authentication shall:

* Include a Nonce element in the first authentication frame to indicate SNonce.
* Include RSNE in the first authentication frame to indicate AKM and pairwise cipher suite.
* Not include a AKM Suite Selector element.
* Include RSNXE in the first authentication frame if any subfield of the Extended RSN Capabilities field in this element is nonzero, except the Field Length subfield.
* Include a Diffie-Hellman Parameter element in the first authentication frame.
	+ Choose indicated finite cyclic group in the Diffie-Hellman Parameter element from the dot11RSNAConfigDLCGroupTable that is at least of the security strength provided by the AKM and cipher suites.
	+ With the chosen finite cyclic group, generate an ephemeral (random) private key, use the selected group’s scalar operation (see 12.4.4.1 (General)) with the private key to generate its ephemeral public key, and indicate the ephemeral public key in the Diffie-Hellman Parameter element.

Otherwise, an authentication originator shall not include a Diffie-Hellman Parameter element or a RSNE or a RSNXE or a Nonce element in the first authentication frame for 802.1X authentication.

For the purpose of interoperability, an authenticator or a supplicant shall support group 19, an ECC group defined over a 256-bit prime order field.

An authentication responder that sets the Encryption of the Frame Body Field of the (Re)Association Request/Response Frame Support subfield in the RSNXE to 1, has the corresponding SME to act as the Authenticator, and receives the first authentication frame with a Nonce element, RSNE, RSNXE, and a Diffie-Hellman Parameter element shall:

* Verify that the AKM indicated in the RSNE rather than AKM suite selector element as defined in 12.4.4 (IEEE 802.1X authentication utilizing Authentication frames) is supported. Otherwise, the authentication responder shall reject message 1 with status code set to STATUS\_INVALID\_AKMP.
* Verify that the pairwise cipher indicated in the RSNE is supported. Otherwise, the authentication responder shall reject message 1 with status code set to STATUS\_INVALID\_PAIRWISE\_CIPHER.
* Validate that finite cyclic group indicated in the Diffie-Hellman Parameter element in the first authentication frame is supported (present in dot11RSNAConfigDLCGroupTable). Otherwise, the authentication responder shall reject message 1 with status code set to UNSUPPORTED\_FINITE\_CYCLIC\_GROUP.
* Verify the public key indicated in the Diffie-Hellman Parameter element in message 1 as specified in 5.6.2.3 of NIST SP 800-56A R2. If verification fails, the authentication responder shall reject the first authentication frame with status code set to INVALID\_PUBLIC\_KEY.
* If the first authentication frame is not rejected, store the indicated SNonce and generate an ephemeral (random) private key with the chosen finite cyclic group and use the selected group’s scalar operation with the private key to generate its ephemeral public key. Perform the group’s scalar-op (see 12.4.4.1 (General)) with the authenticator originator’s ephemeral public key and its own ephemeral private key to produce an ephemeral Diffie-Hellman shared secret, DHss.
* Include a RSNE in the second authentication frame to indicate the AKM and pairwise cipher indicated in the first authentication frame.
* Not include a AKM Suite Selector element in the second authentication frame.
* Include a Diffie-Hellman Parameter element in the second authentication frame.
	+ Indicate chosen finite cyclic group in the Diffie-Hellman Parameter element of the second authentication frame, which is the same as the finite cyclic group in the Diffie-Hellman Parameter element of the first authentication frame.
	+ Indicate its ephemeral public key in the Diffie-Hellman Parameter element of the second authentication frame.
* Include a Nonce element in the second authentication frame to indicate ANonce.

Otherwise, an authentication responder shall not include a Diffie-Hellman Parameter element or a Nonce element or a RSNE in the second authentication frame for 802.1X authentication.

After receiving the second authentication frame with the status code set to SUCCESS, an authentication originator shall:

* If the authentication originator includes a Diffie-Hellman Parameter element in the first authentication frame, validate that there is a Diffie-Hellman Parameter element and a RSNE included in the second authentication frame and there is no AKM suite selector element in the second authentication frame. If the validation fails, the authentication originator shall discard the frame and terminate further protocol processing.
* If the authentication originator does not include a Diffie-Hellman Parameter element in the first authentication frame, validate that there is no Diffie-Hellman Parameter element and no RSNE included in the second authentication frame. If the validation fails, the authentication originator shall discard the frame and terminate further protocol processing.
* If the authentication originator includes a Diffie-Hellman Parameter element in the first authentication frame, validate that the finite cyclic group indicated in the Diffie-Hellman Parameter element in the second authentication frame is the same as the finite cyclic group indicated in the Diffie-Hellman Parameter element in the first authentication frame, validate that the pairwise cipher suite and the AKM indicated in the second authentication frame are the same as the pairwise cipher suite and the AKM indicated in the first authentication frame. The validation of AKM is based on the AKM indication in RSNE rather than AKM suite selector element as defined 12.4.4 (IEEE 802.1X authentication utilizing Authentication frames) If the validation fails, the authentication originator shall discard the frame and terminate further protocol processing.
* Verify the public key indicated in the Diffie-Hellman Parameter element in the second authentication frame as specified in 5.6.2.3 of NIST SP 800-56A R2. If verification fails, the authentication originator shall discard the frame and terminate further protocol processing.
* If the second authentication frame is not discarded, store the indicated ANonce, perform the group’s scalar-op (see 12.4.4.1 (General)) with the authentication originator’s ephemeral public key and its own ephemeral private key to produce an ephemeral Diffie-Hellman shared secret, DHss.

Before sending the authentication frame carrying EAP Success, an authentication responder shall:

* Derive PTK with DHss as defined in 12.7.1.3 (Pairwise key hierarchy).
* Upon completion of PTK generation, the shared secret, DHss, shall be irretrievably deleted.

After receiving the authentication frame carrying EAP Success, an authentication originator shall:

* Derive PTK with DHss as defined in 12.7.1.3 (Pairwise key hierarchy).
* Upon completion of PTK generation, the shared secret, DHss, shall be irretrievably deleted.

The authentication originator and the authentication responder then continue the operation as defined in 12.14.5 (Encryption of the Frame Body Field of the (Re)Association Request/Response Frame) with the following additional rules:

* Authentication responder shall verify that the RSNE in the (Re)Association Request frame is the same as the RSNE included in the first authentication frame. Authentication responder shall also verify that the RSNXE (if present) in the (Re)Association Request is the same as the RSNXE included in the first authentication frame. If the validation fails, the authentication responder shall reject the association.
* Authentication originator shall verify that the RSNE in the (Re)Association Response frame is the same as the RSNE included in the second authentication frame. If the validation fails, the authentication originator shall disassociate.

**TGbi Editor: *Instruction: Modify 12.7.1.3 (Pairwise key hierarchy) as shown below***

**12.7.1.3 Pairwise key hierarchy**

(…existing texts….)

The following apply when not using FILS authentication:

* SNonce is a random or pseudorandom value contributed by the Supplicant; its value is taken when a PTK is instantiated and is sent to the PTK Authenticator.
* ANonce is a random or pseudorandom value contributed by the Authenticator.
* The PTK shall be derived from the PMK by

PTK = (#3686)PRF-*Length*(PMK, “Pairwise key expansion”, Min(AA,SPA) || Max(AA,SPA) || Min(ANonce,SNonce) || Max(ANonce,SNonce)||DHss) if key derivation with Authentication frame exchange for 802.1X is used as defined in 12.14.7.2 (802.1X).

Otherwise, PTK = (#3686)PRF-*Length*(PMK, “Pairwise key expansion”, Min(AA,SPA) || Max(AA,SPA) || Min(ANonce,SNonce) || Max(ANonce,SNonce))

(…existing texts….)