IEEE P802.11  
Wireless LANs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Draft Text for 12.15 Security | | | | |
| Date: 2020-01-xx | | | | |
| Author(s): | | | | |
| Name | Affiliation | Address | Phone | email |
| Hitoshi Morioka | SRC Software | Fukuoka, JAPAN |  | [hmorioka@src-soft.com](mailto:hmorioka@src-soft.com) |
|  |  |  |  |  |

Abstract

This document describes a TGbc draft text proposal for clause 12 security.

*Yellow marked numbers are temporal and to be assigned by ANA.*

12. Security

*Add the following new subclause in clause 12.*

12.15 Frame authentication for eBCS

12.15.1 General

eBCS provides one-way frame authentication mechanisms that do not require key negotiation between a transmitter and receivers.

eBCS uses one of two types of frame authentication.

One is eBCS Public Key Frame Authentication (PKFA) and the other is eBCS Hash Chain Frame Authentication (HCFA).

While the PKFA is suitable for occasional small data transfer, the HCFA is suitable for continuous content distribution such as live streaming or periodic file transfer.

The following preparations are required for both authentication scheme before starting eBCS.

* The certificate(s) of the CA(s) (Certificate Authority) shall be installed into the eBCS receivers.
* The eBCS transmitter generates its own private key and public key pair. The public key shall be signed by one of the CAs of which the eBCS receiver have the certificate.

12.15.2 eBCS public key frame authentication (PKFA)

12.15.2.1 Signing

One of the following public key algorithms is used.

* RSA-2048
* DSA-L2048N256
* ECDSA-P256

RSA-2048 computational cost for signing is high and for verification is low. This algorithm is good for downlink use. DSA-L2048N256 computational cost for signing is low and for verification is high. This alogorithm is good for uplink use. ECDSA-P256 computational cost for signing and for verification are middle.

The eBCS transmitter generates an eBCS Info frame when it receives data to be transmitted. The eBCS Info frame contains the following items.

* eBCS Info sequence number
* Timestamp
* Authentication algorithm
* Allowable time difference
* Certificate length
* Certificate
* Contents Information
* Signature

Fill all the fields according to 11.55.2.1 (eBCS Info frame transmission) except the signature.

generate signature as following:

Signature = Sign(The eBCS transmitter’s private key, SHA3-256(Transmitter’s MAC address | from the Sequence Number field to the last Contents Information field in the eBCS Info frame))

If the length of the eBCS Info frame is larger than the maximum MMPDU length (Table 9-25 Maximum data unit sizes (in octets) and durations (in microseconds)), the eBCS Info frame shall be fragmented as described in 11.55.\* (eBCS Info fragmentation).

Then the eBCS transmitter transmits the eBCS Info frame.

12.15.2.2 Verification

When the eBCS receiver receives the eBCS Info frame, the eBCS receiver shall authenticate it as following:

1. If the eBCS Info frame is fragmented, defragment it at first as described in 11.55.\* (eBCS Info defragmentation).
2. If the difference between the timestamp in the eBCS Info frame and the time of the eBCS receiver is greater than the allowable time difference in the eBCS Info frame, the eBCS Info frame shall be discarded.
3. Verify the certificate in the eBCS Info frame by the installed CA’s certificate. If the verification is failed or the certificate of the CA that signed the certificate in the eBCS Info frame is not installed, the eBCS Info frame shall be discarded.
4. Verify the signature in the eBCS Info frame by the certificate in the eBCS Info frame. If the verification is failed, the eBCS Info frame shall be discarded.

If the authentication succeeds, the eBCS receiver processes the Data in the Contents Information field(s) in accordance with 11.55.2.2 (eBCS Info frame reception).

12.15.3 eBCS hash chain frame authentication (HCFA)

12.15.3.1 General

The HCFA uses the digital signature and the modified TESLA (Timed Efficient Stream Loss-Tolerant Authentication, IETF RFC4082).

TESLA is a kind of oneway key chain authentication mechanism. The eBCS transmitter generates TESLA base keys and TESLA authentication keys for each content stream before each eBCS Info frame generation. One of the following hash functions are used for TESLA key generation.

* SHA3-256
* SHA3-384

The TESLA base keys (*Bn*) are generated as following:

*Bs,0* = Random value (length = the length of the selected hash function output)

*Bs,n* = Hash(“eBCS TESLA base key” || *Bs,n-1*) (*n* >= 1)

where *s* is the sequence number of the generating eBCS Info frame.

The TESLA authentication keys (*An*) are generated as following:

*As,n* = Hash(“eBCS TESLA authentication key” || *Bs,n*)

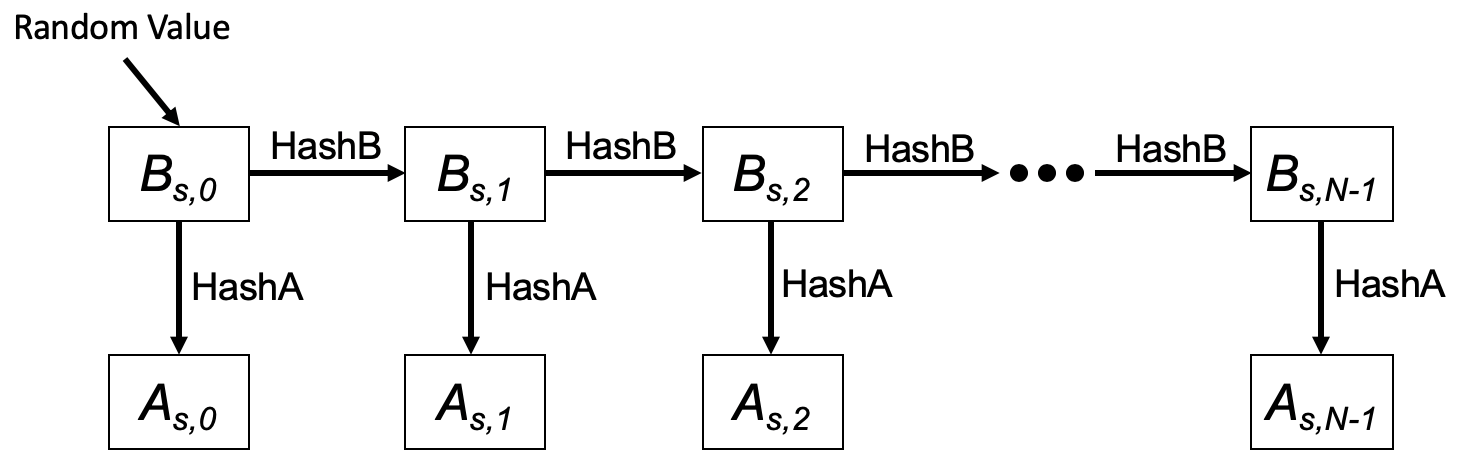
where *s* is the sequence number of the generating eBCS Info frame.

The number of the keys to be generated (*N*) is following:

*N* = *TI* / *TK* + 3

where *TI* is the eBCS Info frame transmission interval and *TK* is the TESLA key change interval.

The TESLA key generation scheme is shown in Figure 12-bc1 (TESLA key generation scheme).



**Figure 12-bc1 TESLA key generation scheme**

In this figure, HashB is the hash function to generate TESLA base keys and HashA is the hash function to generate TESLA authentication keys.

The HCFA provides the following two authentication methods to authenticate each eBCS Data frame.

* Instant authentication
* TESLA authentication

Each authentication method uses separate authenticator. The instant authentication uses the instant authenticator, and the TESLA authentication uses the TESLA authenticator.

The instant authenticator is the hash value of the eBCS Data frame to be transmitted later that is generated as following:

Instant Authenticator = Hash(eBCS Data frame to be transmitted later)

/\* details are to be defined after the consensus about eBCS Data frame \*/

The TESLA authenticator is the HMAC-hash value of the eBCS Data frame that contains the TESLA authenticator with TESLA authentication key.

TESLA Authenticator = HMAC-Hash(*An*, eBCS Data frame including Instant Authenticator)

/\* details are to be defined after the consensus about eBCS Data frame \*/

The HCFA uses both eBCS Info frames (9.6.7.52 eBCS Info frame format) and eBCS Data frames (/\* reference to be added \*/). The frame sequence is shown in Figure 12-bc2 (eBCS HCFA frame sequence).

スクリーンショット が含まれている画像

自動的に生成された説明

**Figure 12-bc2 eBCS HCFA frame sequence**

The eBCS Info frames are transmitted periodically in the interval of dot11EBCSInfoInteval (*TI*). *TK* is the TESLA key change interval configured as dot11EBCSTeslaKeyChangeInterval. *TI* shall be a multiple of *TK*. The period between one eBCS Info frame and the next eBCS Info frame is called “HCFA period”. Each HCFA period is identified by the HCFA sequence number.

Each content stream has its index that is determined by the order, starts with index 0, in the Contents Information field in the eBCS Info frame.

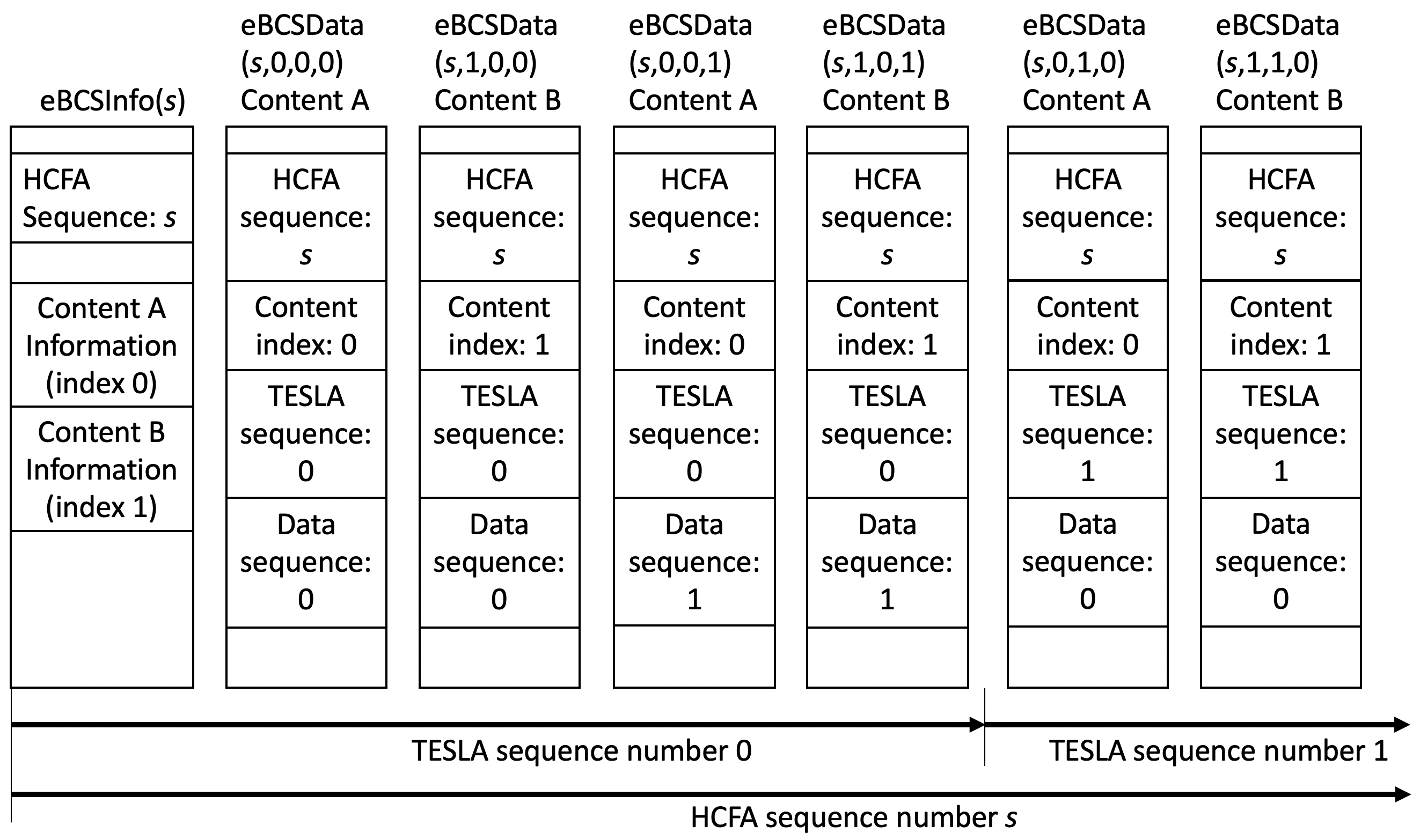
The period that uses the same TESLA authentication key is called “TESLA period”. Each TESLA period has its sequence number, TESLA sequence number, that starts with 0 at the beginning of each HCFA period. Note that the TESLA sequence number is different from TESLA key indexes.

Each eBCS Data frame has its sequence number that starts from 0 at the beginning of each TESLA period. The eBCS Data frame is identified by the following identifiers:

* HCFA sequence number
* Content stream index
* TESLA sequence number
* Data sequence number

eBCSData(*s*, *c*, *t*, *d*) represents the eBCS Data frame that the HCFA sequence number is *s*, the Content stream index is *c*, the TESLA sequence number is *t* and the Data sequence number is *d*. IAuth(*s*, *c*, *t*, *d*) and TAuth(*s*, *c*, *t*, *d*) represent the instant authenticator and the TESLA authenticator for the eBCSData(*s*, *c*, *t*, *d*) respectively. eBCSInfo(*s*) represents the eBCS Info frame that the HCFA sequence number is *s*.

For example, in case of two contents stream, Content A and Content B, the identifiers are shown in Figure 12-bc3 (Identifiers example).



**Figure 12-bc3 Identifiers example**

The index of TESLA base key and TESLA authentication key is defined as B(s, c, t) and A(s, c, t) respectively where s is the HCFA sequence number, c is the content index, t is the TESLA sequence number. The TESLA sequence number is different from the TESLA base/authentication key index. The TESLA base/authentication keys are used in the opposite order of the TESLA key generation. The relation between the TESLA base/authentication key index and the TESLA sequence number is shown in Table 12-bc1 (Relation between TESLA authentication key index and TESLA sequence number) where N is the number of the TESLA authentication key.

**Table 12-bc1 Relation between TESLA base/authentication key index and TESLA sequence number**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| TESLA base/authentication key index | (N-1) | (N-2) | (N-3) | N-4 | N-5 | N-6 | … | 0 |
| TESLA sequence number | (-3) | (-2) | (-1) | 0 | 1 | 2 | … | N-4 |

Note: TESLA sequence number -3, -2 and -1 are used only for key verification.

12.15.3.2 eBCS Info frame generation

The eBCS Info frame contains the following items that are related to frame authentication.

* HCFA sequence number
* Timestamp
* Certificate
* Signature
* TESLA key change interval
* Contents Information
  + TESLA base key(s)
  + Instant authenticator(s) of eBCS Data frames to be transmitted in next 2 \* *TK*

The functions of the eBCS Info sequence number, the timestamp, the certificate and the signature are same as those of PKFA.

The TESLA key change interval, TESLA base key(s) and the instant authenticator(s) are present only in HCFA.

The TESLA key change interval is *TK*.

The TESLA base keys to be includes in the eBCS Info frame of the sequence number *s* are B(*s*, *c*, -3), B(*s*-1, *c*, 1) and B(*s*-1, *c*, 0) for all content streams, where c is the content index and N is the number of TESLA authentication keys. In case of the first eBCS Info frame, B(*s*-1, *c*, 1) and B(*s*-1, *c*, 0) are not present.

For generating the instant authenticators, the eBCS transmitter requires to buffer data packets during 2 \* *TK*.

If the length of the eBCS Info frame is larger than the maximum MMPDU length (Table 9-25 Maximum data unit sizes (in octets) and durations (in microseconds)), the eBCS Info frame shall be fragmented as described in 11.55.\* (eBCS Info fragmentation).

12.15.3.3 eBCS Data frame generation

/\* this subclause will be updated after consensus about eBCS Data frame type \*/

The eBCS Data frame contains the following items.

* Content data
* HCFA sequence number
* Content index
* TESLA sequence number
* Data sequence number
* TESLA base key
* Instant authenticator(s)
* TESLA authenticator

The HCFA sequence number, Content index, TESLA sequence number and Data sequence number are described in 12.15.3.1 (General).

The TESLA base key contained in eBCSData(*s*, *c*, *t*, *d*) is B(*s*, *c*, *t*-2).

The instant authenticator in the eBCSData(*s*, *c*, *t*, *d*) is IAuth(*s*, *c*, *t*+2, *d*) except the following conditions.

If *t* is larger than N-6 where N is the number of TESLA authentication keys, the instant authenticator is not included in the eBCS Data frame. When DataN(*s*, *c*, *t*) represents the number of eBCS Data frames in HCFA sequence number *s*, content stream *c* and TESLA period *t*, if DataN(*s*, *c*, *t*) is larger than DataN(*s*, *c*, *t*+2), the last DataN(*s*, *c*, *t*) - DataN(*s*, *c*, *t*+2) eBCS Data frames does not contain the instant authenticator. If DataN(*s*, *c*, *t*) is smaller than DataN(*s*, *c*, *t*+2), the eBCSData(*s*, *c*, *t*, DataN(*s*, *c*, *t*)-1) contains IAuth(*s*, *c*, *t*+2, *d*)…IAuth(*s*, *c*, *t*+2, DataN(*s*, *c*, *t*+2)-1).

The TESLA authenticator in the eBCSData(*s*, *c*, *t*, *d*) is TAuth(*s*, *c*, *t*, *d*) with A(*s*, *c*, *t*).

12.15.3.4 eBCS Info frame reception

The received eBCS Info frame, eBCSInfo(*s*), is processed as following.

1. If the eBCS Info frame is fragmented, defragment it at first as described in 11.55.\* (eBCS Info defragmentation).
2. If the diference between the timestamp in the eBCS Info frame and the time in the eBCS receive’s clock is greater than the TESLA key change interval in the eBCS Info frame, the eBCS Info frame shall be discarded.
3. Verify the certificate in the eBCS Info frame by the installed CA’s certificate. If the verification is failed or the certificate of the CA that signed the certificate in the eBCS Info frame is not installed, the eBCS Info frame shall be discarded.
4. Verify the signature in the eBCS Info frame by the certificate in the eBCS Info frame. If the verification is failed, the eBCS Info frame shall be discarded.
5. If the TESLA base key(s) of the previous HCFA period, B(*s*-1, *c*, *N*-4) and B(*s*-1, *c*, *N*-5), is contained and the eBCS Data frames of the previous HCFA period to be authenticated are present, authenticate and process the eBCS Data frames as described in 12.15.3.5 (eBCS Data frame reception).
6. Cache the TESLA base key(s), B(*s*, *c*, 0), for the HCFA period of the eBCS Info frame.
7. Cache the instant authenticaters contained in the eBCS Info frame.

12.15.3.4 eBCS Data frame reception

/\* this subclause will be updated after consensus about eBCS Data frame type \*/

The eBCS Data frame shall be discarded until the eBCS Info frame from the BSS is received.

The received eBCS Data frame, eBCSData(*s*, *c*, *t*, *d*), is processed as following.

1. Compute B(*s*, *c*, *t*-1) from B(*s*, *c*, *t*) in eBCSData(*s*, *c*, *t*, *d*). If the computed B(*s*, *c*, *t*-1) is different from the cached B(*s*, *c* , *t*-1), The eBCSData frame shall be discarded.
2. If the instant authenticator of the eBCS Data frame, IAuth(*s*, *c*, *t*, *d*), is cached, compare the hash value computed by the eBCS receiver and the cached instant authenticator. If the computed hash value is different from the cached instant authenticator, the eBCS Data frame shall be discarded.
3. If the instant authenticator of the eBCS Data frame, IAuth(*s*, *c*, *t*, *d*), is not cached, the eBCS Data frame may be cached until the TESLA base key for the TESLA period is received, or the eBCS Data frame may be discarded.
4. If the eBCS Data frame(s) using the TESLA authentication key derived from the TESLA base key included in the eBCS Data frame is cached,
5. Derive the TESLA authentication key from the TESLA base key.
6. Compute TESLA authenticator for the cached eBCS Data frame by using the TESLA authentication key.
7. If the computed TESLA authenticator is different from the TESLA authenticator in the cached eBCS Data frame, the cached eBCS Data frame shall be discarded.

Then forward the content in the eBCS Data frame to the higher layer.