**3GPP TSG SA WG3 Security — S3#29** **S3-030452**
**15 - 18 July 2003**
**San Francisco, USA**

*3GPP2 S.P0083*
*Version 0.6*
*Version Date: 5 June, 2003*


# Broadcast-Multicast Service Security Framework

***EDITOR***

Philip Hawkes
QUALCOMM Australia
+(61-2) 9817-4188
phawkes@qualcomm.com

***REVISION HISTORY***

| ***REVISION HISTORY*** | | |
|---|---|---|
| ***Rev. 0.1*** | *First draft* | *11 September 2002* |
| ***Rev. 0.2*** | *Second draft: Simplified description to make it less architecture-dependant.* | *28 October 2002* |
| ***Rev. 0.3*** | *Third draft: Merged Lucent and Qualcomm proposals. This version was approved to be the baseline text at conference call 2002/11/19.* | *19 November 2002* |
| ***Rev. 0.4*** | *Fourth draft: major changes and clarifications.* | *11 December 2002* |
| ***Rev 0.5*** | *Title changed from "Broadcast Security Framework" to "Broadcast-Multicast Service Security Framework".* | *13 January 2003* |
| ***Rev 0.6*** | *Terminology changes to reflect decisions in TSG-X BCMCS Ad Hoc. Also some minor editorial changes.* | *5 June 2003* |
| ***Rev 0.7*** | *BCMCS Security Algorithhms specified ( Section 4.5)* | *15 July 2003* |

# Table of Contents

# 1 Introduction and Scope

This document defines the security framework for the Broadcast-Multicast Services (BCMCS). The security framework provides a logical description of the security information, functions and protocols for BCMCS. The architectural design of the network that supports these functions is outside the scope of this document.

# 2 References

S.S0053

S.S0055

# 3 Definitions and Abbreviations

**BAK** 128-bit Broadcast Access Key: Provides access to one or more multicast IP flows of a particular set of BCMCS programs for a certain amount of time (for example, one day, week or month). Each encrypted set of BCMCS programs have a different BAK value. Each BAK should have the following associated values:

**BAK_ID:** BAK identifier. A sequence number that identifies which value of BAK is currently valid for a particular BCMCS Multicast IP Flow; that is, the BAK is identified by the combination of a BCMCS_FLOW_ID and BAK_ID. For a particular BAK, the corresponding value of BAK_ID is the same for all users.

**BAK_Expire:** Indicates when the BAK will expire. This may be indicated by the time when the BAK will expire, or by the Delta time left until the BAK expires, in which case it is calculated when a new BAK is received. Therefore, there may be multiple BAK_Expire values associated with the same BAK. Before the BAK_Expire time expires, the MS is expected to request a new BAK value.

**BCMC Content** Broadcast-Multicast Content, also referred to simply as **Content**. The content is the data being broadcast. Typically, this is expected to be audio-visual data.

**BCMCS** Broadcast-Multicast Service. BCMCS is the name of the system offering broadcast and multicast services.

**BCMCS Content Stream** A single BCMCS program identified by content name. This is also referred to simply as a **content stream.** A content stream can be considered as equivalent to a TV channel. For example, the document may refer to a “CNN BCMCS content stream” or “local news BCMCS content stream”. A BCMCS Content Stream may consist of multiple Multicast IP Flows.

**BCMCS_FLOW_ID** A value used for identification of a BCMCS Multicast IP Flow.

**CS** Content Source (functional entity): Provides the unencrypted data for the service. A CS may be a proxy or an aggregator for other content servers, but from a network point of view, it is the provider of content to the MS.

**BAKG** BAK Generator (functional entity): Generates BAK, determines BAK_ID and BAK_Expire, and delivers them to the BAKD and SKM. BAKs should be generated in a secure manner and appear random.

**BAKD** BAK Distributor (functional entity): Obtains authorization and TK from SM and encrypts BAK for delivery to UIMs.

**CE** Content Encryptor (functional entity): Encrypts content from the CS, and sends the encrypted content to the MS via the cellular system.

**MS** Mobile Station: For the purposes of this document, the MS is considered as two separate entities, the UIM and ME. Note that this does not preclude the use of a stand-alone secure MS (trustworthy mobile equipment with internal secure memory).

**UIM**: User Identity Module (UIM): The UIM is a low power processor that contains secure memory. The UIM may be removable (like a SIM card) or part of the MS itself. The UIM calculates the SK used by the ME to decrypt the content.

**ME**: Mobile Equipment: The ME contains a high power processor, but no secure memory. It uses the SK, received from the UIM, to decrypt the content.

**Multicast IP Flow**. Similar to an ordinary IP flow, with the exception that the destination address is an IP Multicast address. The flow can be identified by source address, source port, destination IP Multicast address, and destination port.

**RK** Registration Key: provisioned in the UIM and SM prior to service. Each SM should have a unique 128-bit RK for each UIM.

**SK** Short-term Key: The 128-bit SK is used by the CE to encrypt the BCMC content and the SK is used by the MS to decrypt the content. The SK is calculated in the SKM and the UIM. The SK is derived from SK_RAND and BAK.

**SK_RAND**: SK Random Number used to calculate a unique SK.

**SKM** SK Manager (functional entity): Generates SK using BAK and SK_RAND and passes SK, SK_RAND and BAK_ID to the CE.

**SP** Service Provider: a network providing voice/data services.

**Visited SP** The serving network in which the MS is currently located.

**Home SP** The network holding the user's voice/data subscription.

**SM** Subscription Manager (functional entity): performs accounting, authentication and authorization for BCMCS. The SM also calculates the TK, based on the RK, used to hide the BAK. The SM may be the subscriber's home AAA (H-AAA) or be an independent entity.

**TK** 128-bit Temporary Key: used by the BAKD to encrypt BAK when provisioning BAK in the UIM. The TK is obtained from the SM.

# 4 Overview of BCMCS

## 4.1 Introduction to BCMCS

The aim of BCMCS is to provide broadcast/multicast service to authorized (subscribed) users. The Content Sources (CSs) provide content, via a cellular serving system, to the BCMCS subscribers. The content may be IP multimedia messages such as audiovisual data or broadcast multimedia messages. A CS may be part of the serving network, or an independent entity. If the content is offered for free, then any user may view/process the content. If access to the BCMCS is subscription based, then the content can be encrypted so that only the authorized users can view/process the content. This document addresses the security needs for BCMCS.

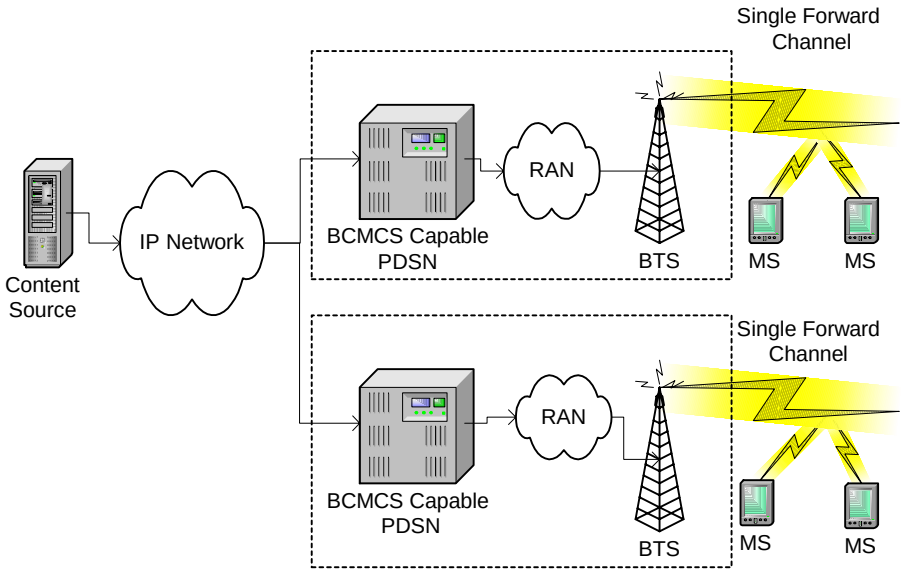


Figure 4.1 High Level view: Broadcast/Multicast Service

A single CS may provide more than one BCMCS to one or more carriers. The users may subscribe to services through their own carrier, or directly from a CS. The entity that supports the user subscription profile is the *Subscription Manager* (SM). The user may subscribe to more than one service through one or more SMs. The SMs provide service authorization and are an integral part of the BCMCS key management.

## 4.2 Summary of Key Management

For the purposes of this document, the Mobile Station (MS) is considered as two separate entities: the User Identity Module (UIM) and the Mobile Equipment (ME). The UIM is a low power processor that contains secure memory. The ME contains a high power processor, but no secure memory. The ME and the UIM may be integrated within one physical unit. In fact an MS can even be a trustworthy ME with its own internal secure memory.

The user authentication and authorization for cellular services, as well as network security is outside the scope of this document. This document assumes the following trust model: all network entities are presumed to trust the other network entities that they communicate with, all communication between network entities is presumed to be secure, and all network entities are presumed to perform their tasks correctly. The UIM is trusted by the SM and vice versa. All network entities trust the UIM to keep secrets and perform its tasks correctly. However, the UIM trusts only the SM.

The main threat addressed by this document is the threat of a user obtaining cheap and reliable access to the content of a particular content stream without being authorized for that particular content stream. To counter this threat, the content is delivered encrypted to the ME, and the decryption keys are provided only to those users who are authorized (subscribed) to receive

BCMCS. The primary focus of this document is the key management scheme for content streams that require authorization.

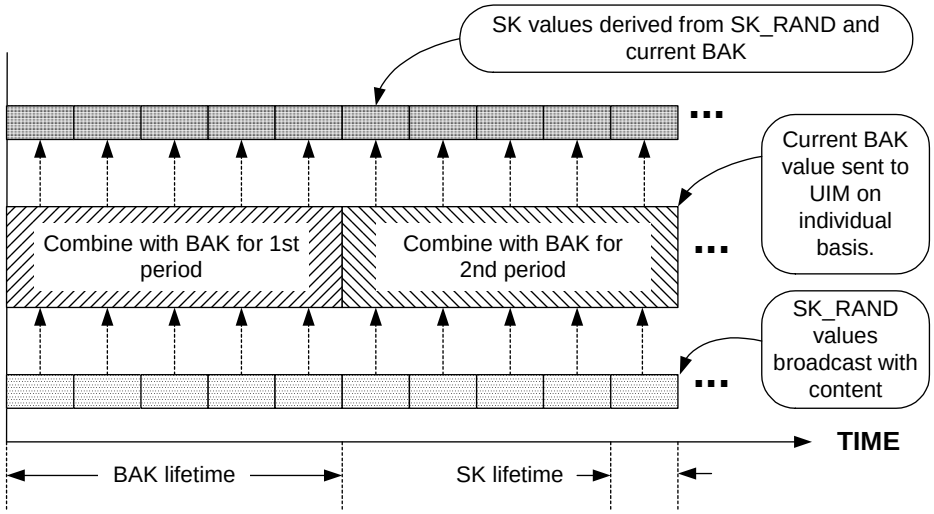


Figure 4.2. BCMCS Key Management. Multiple SK values are generated for each BAK value. The SK values are generated by combining BAK with the SK_RAND value that is transmitted with the encrypted content.

The content is encrypted using a unique and frequently changing *Short-term Key* (SK). The ME decrypts the content using the same SK. The SK should be frequently changed to minimizing the impact of a "rogue shell" sharing its SK with unauthorized users. The SK is never transmitted over the air; it is derived by the UIM from a *Broadcast Access Key* (BAK) and a random value broadcasted along with the encrypted content.

In order for the user to decrypt the necessary BAK values, the user's UIM must share an RK with an SM. Provisioning of RK is outside the scope of this document. A temporary encryption key (TK) is derived from the RK by the SM, sent to the BAKD, and subsequently used by the BAKD to encrypt/decrypt the BAK.

The BAK is encrypted by the BAK Distributor (BAKD), which may be associated with the Content Source, with the Subscription Manager, or the visited cellular system. The same BAK is provided to all users granted access to a particular Multicast IP Flow, and it is active for a pre-defined period of time. The multiple Multicast IP Flows associated with the particular BCMCS content stream may share the same BAK. Once the UIM has obtained the encrypted BAK (from the BAKD), it recovers the BAK (with its own calculated TK), and then computes the SK value needed to decrypt the Multicast IP Flow. The SK is then delivered to the ME for content decryption. The BAK is never divulged to the ME, and only the UIM with a correctly pre-provisioned RK can recover the BAK.

## 4.3 Security Functional Architecture

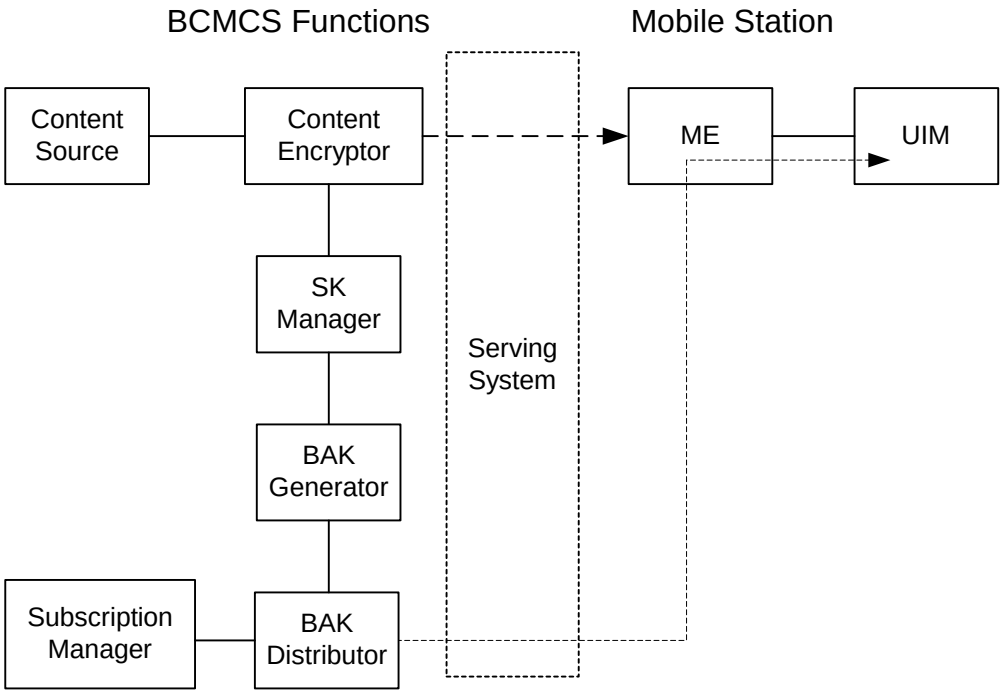


Figure 4.3. Functional Architecture

Figure 4.3 shows the functional entities that may be involved in BCMCS. Those entities represent functions essential to support secure BCMCS. They could be incorporated in one or more physical entities in the network. It should not be assumed that these functional entities correspond to separate architectural entities, so not all interfaces will require standardization. The allocation of the security functions to network entities is outside the scope of this document.

The BCMCS functional entities are described in more detail below.

### 4.3.1 Summary of Functional Entities

**CONTENT PATH**

**Content Source (CS)**: Generates the un-encrypted content.

**Content Encryptor (CE)**: Encrypts the content using the SK provided by the SKM. Encryption may take place at the Content Source or in the serving cellular system.

**KEY/AUTHORIZATION PATH**

**Subscription Manager (SM)**: May provide the functions of Authentication, Authorization and Accounting (AAA). The SM shares a Registration Key RK with the UIM: This key RK may be the A-key (which is the basis of key distribution and authentication for voice/data services as described in S.S0053), the key K used for AKA (as described in S.S0055), or some other key provisioned specifically for BCMCS. The SM calculates the TK, based on the user specific RK. *The provisioning of RK in the UIM and SM is beyond the scope of this document.*

**BAK Generator (BAKG):** Generates the Broadcast Access Key (BAK) as required, and distributes BAK and to BAKD and SKM.

**BAK Distributor (BAKD):** Controls the distribution of the Broadcast Access Key (BAK).

**SK Manager (SKM):** Controls the updating and distribution of the Short-term Key (SK).

**Mobile Station (MS)** For the purposes of this document, the MS is considered as two separate entities, the UIM and ME.

**UIM**: User Identity Module : The UIM shares a key RK with the SM. The UIM performs all key management related to BCMCS.

**ME**: Mobile Equipment. Includes equipment for receiving the BCMCS. The ME performs decryption of encrypted content using SK obtained from the UIM.

### 4.3.2 Summary of Key Distribution

Figure 4.4 shows the basic communications involved in the key distribution for BCMCS encryption. Many details are omitted in this diagram for the sake of clarity. The Figure is described below.

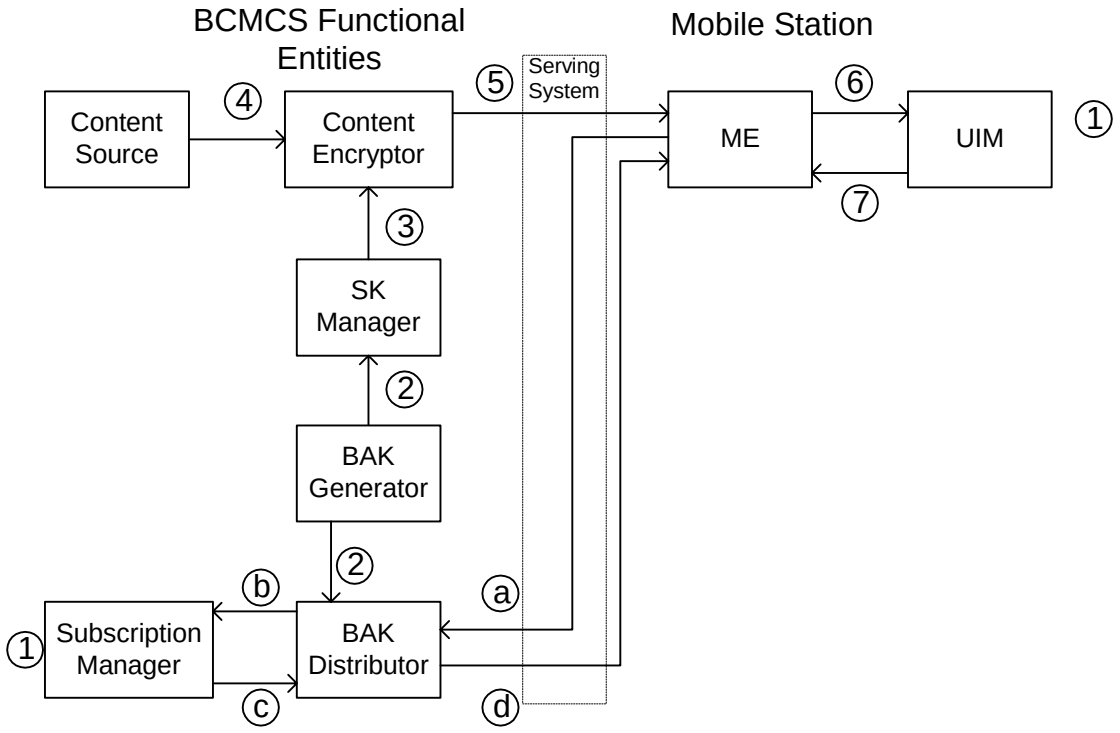


Figure 4.4 Logical Architecture showing functional entities involved in BCMCS security.

#### 4.3.2.1 BAK Generation

The following steps (1-2) are performed prior to the transmission of encrypted Multicast IP Flow(s) to the MS:

1. The UIM and SM are provisioned with a Registration Key RK that will be the basis of authentication and key exchange with respect to BCMCS.

2. The BAKG generates a value for BAK, and associates the value with an identifier BAK_ID and an expiry time (BAK_Expire). The value of BAK, along with the corresponding values of BAK_ID and BAK_Expire are passed to the SKM and BAKD.

#### 4.3.2.2 Normal Procedure for Encrpting a Multicast IP Flow With Unchanged BAK

The following steps (3-7) occur under normal circumstances, when BAK is unchanged from the previous decryption operation in the MS.

3. The SKM creates SK from the current BAK and a random value SK_RAND. The SKM passes SK, SK_RAND, BAK_ID and BAK_Expire to the CE.

4. The CS sends Multicast IP Flow to the CE.

5. The CE encrypts the Multicast IP Flow using SK and sends the encrypted Multicast IP Flow to the MS via the serving system. The CE also includes SK_RAND and BAK_ID with the encrypted Multicast IP Flow.

6. The ME receives the encrypted Multicast IP Flow, and takes action as follows:

   a. If BAK_ID and SK_RAND are unchanged from the last received Multicast IP Flow, the ME decrypts the Multicast IP Flow using the value of SK currently assigned to that Multicast IP Flow and passes the result to the user application;

   b. If BAK_ID or SK_RAND have changed, the ME requests a new SK from the UIM, including the BCMCS_FLOW_ID, BAK_ID and SK_RAND.

7. The UIM generates SK from BAK and SK_RAND and returns SK to the ME, which decrypts the Multicast IP Flows and passes the result to the user application.

#### 4.3.2.3 Procedure With Changed or Unavailable BAK

Steps (a-d) in Figure 4.4 are performed to request a new BAK. Section 4.4 discusses possible methods by which the MS determines that a new BAK is needed.

a. The ME sends a BAK request to the BAKD. . The BAK request may include authentication information based on RK, which can be used by the SM to determine that the request came from a legitimate subscriber. Section 4.4.1 discusses the need for authentication of BAK requests.

b. In order to send BAK to the UIM, the BAK must be encrypted to protect it against reception by other than the intended recipient. The BAKD requests a temporary key (TK) generated by the SM.

c. The SM generates TK from a random value TK_RAND and RK. TK_RAND may be generated by the BAKD, or by the SM. TK_RAND may also be used as a challenge in the authentication process described in step a. The SM sends TK and TK_RAND to the BAKD.

d. The BAKD encrypts the value of BAK with TK, and sends the encrypted BAK, along with TK_RAND and BAK_Expire to the UIM via the ME. The UIM first forms TK from TK_RAND and RK, and then decrypts the encrypted BAK with TK to form BAK. The value of BAK and its associated BAK_Expire are stored in the UIM. The UIM should be able to store at least two values of BAK so that a new BAK can be obtained and stored in anticipation of the BAK expiration time.

## 4.4 BAK Management

A user can repeat the BAK update process with multiple content streams. Thereafter (from a security perspective) the MS has the potential to be managing the decryption keys for multiple content streams simultaneously. In order to determine which BAK corresponds to which Multicast IP Flow, a BCMCS flow identifier BCMCS_FLOW_ID should be stored with the BAK.

This document does not specify how an MS determines that it needs to update BAK. We assume that a means will be provided for the MS to determine that its BAK is about to expire or has expired, triggering action to perform a BAK update. Several methods are possible for accomplishing this. The specific method used is not important for the present subject.

**BAK_ID**. The UIM can determine if a BAK is associated with a particular Multicast IP Flow by referring to the corresponding BCMCS_FLOW_ID. However, there will be times when the UIM needs to store two BAK values associated with a particular BCMCS_FLOW_ID. The reason is that the BAK must be updated in the UIM prior to the BAK actually being used to encrypt/generate SK values (see Section 4.4.3). Consequently, both the old BAK and the new BAK will reside in the UIM simultaneously. The UIM must have some means to determine which BAK is valid. The recommended method is to allocate a BAK identifier BAK_ID to each BAK, in addition to the BCMCS_FLOW_ID. The BAK is identified by the BCMCS_FLOW_ID and BAK_ID. The CE will include the BAK_ID with the encrypted content in the broadcast data. The UIM can now distinguish if the old BAK is still being used or if the new BAK has begun to be used.

A value for BAK_Expire will be provided along with the BAK, so the MS can update expired keys.

**BAK lifetime.** The BAKG decides how often BAK is changed. The following issues should be considered in deciding how often BAK is to be changed.

- Frequent BAK changes will provide more security.

- Frequent BAK changes will also provide greater flexibility in subscription control. We show this by example. Once a user has BAK, they can access the content for the lifetime of that BAK. Suppose the BAK is changed at the beginning of every month. If a user's subscription runs out halfway through the lifetime of a BAK, the user will still be able to generate SK (and thus view the content) until the BAK expires. So by changing BAK only every month, the SM can only charge subscriptions from the beginning of the month to the end of the month. A user can't subscribe from the middle of one month to the middle of the next. However, if BAK changed every day, then the user could subscribe from the beginning of any day during the month.

- Increasing the frequency of BAK changes should be evaluated against a possible increase in the number of times the mobile station has to retrieve new BAK values.

### 4.4.1 Authenticating BAK Requests

An adversary cannot obtain BAK by performing a BAK request while impersonating a subscribed user. Only the subscribed user will be able to derive TK from RAND_TK, and thus extract BAK. For this reason, the BAKD does not need to authenticate BAK requests in order to protect BAK.

If, however, other control functions or accounting data are included with the BAK request, then it is necessary to prove the authenticity of the data source. It may also be desirable to authenticate requests to prevent malicious generation of excess network traffic. *The procedures for authenticating the request are for further study.*

Note that even though authentication of BAK requests may not be needed, there is still a need for authorization of the request. Such authorization must be specific to the BCMCS to which the requested BAK applies.

### 4.4.2 Storage of BAK

It is essential to the security model used herein that the UIM does not reveal BAK. If a single UIM reveals BAK, then all security is compromised until the BAKG changes BAK.

The UIM should store BAK and related data about BAK, such as BAK_ID and expiration time, if any. The BAK is identified by the BCMCS_FLOW_ID (of the corresponding BCMCS Multicast IP Flow) and BAK_ID.

The ME may store the BAK-related data, to save requesting this information from the UIM.

### 4.4.3 Updating BAK before use

It may prove beneficial to provision UIM with BAK shortly before BAK begins being used to derive SK values. Otherwise, once the CE starts sending packets with SK derived from the new BAK, the user would experience a delay as the MS performs a BAK update. If many users are tuned in, then there will be a burst of traffic as all the MSs perform a BAK update.

To avoid such problems, the BAKD should allow an MS to obtain the new BAK shortly before the BAK changes. Different MS may have different schedules for performing BAK updates, to prevent too many MSs performing a BAK update at once.

For security reasons, BAK should be distributed as close as possible to the time of use.

## 4.5 BCMCS Security Algorithms

### 4.5.1 Encryption of Content

[Editor's note: We prefer that the encryption algorithm be based on AES. However the exact mechanism depends on the protocol layer at which the encryption is performed, and this has not yet been decided.]

#### 4.5.1.1 Content Encryption Using Upper Layer protocol

The content of BCMCS shall be encrypted by the upper layer protocols as defined by IETF, protocols capable of delivering BCMCS content. One such protocol is described in the SRTP draft (draft-ietf-ave-srtp-08). The stream encryption key is the Session Key (SK) defined below.

#### 4.5.1.2 Content Encryption Using Link Layer Encryption

TBD.

### 4.5.2 Encryption of BAK

BAK encryption (under key TK) shall be performed using the ESP_AES algorithm as defined in S.S0055.

The input parameter of ESP_AES shall be set as follows:

- The encryption key parameter shall be set to TK.
- The *fresh* parameter shall be set to TK_RAND.
- The *freshsize* parameter shall be set to 8.
- The *buf* parameter shall be set to the address of the octet containing the first bit of the buffer that contains the data to be encrypted or decrypted.
- The *bit_offset* parameter shall be set 0.
- The *bit_count* parameter shall be set to 128.

### 4.5.3 Management of TK

#### 4.5.3.1 TK_RAND

TK_RAND shall be 64 bits in length.

TK_RAND shall be generated by invoking the algorithmic function *f0* as specified in Section 2.2.2.2 of S.S0055.

The input parameters of f0 shall be set as follows:

- The K parameter shall be set to RK.
- The fi parameter shall be set to 0x41.
- The Fmk parameter shall be set to 0x41484147.

TK_RAND shall be set to the computed 64-bit output of *f0*.

#### 4.5.3.2 TK Generation

TK shall be 128 bits in length.

TK shall be generated by invoking the algorithmic function *f3* as specified in Section 2.2.2.8 of S.S0055.

The input parameters of *f3* shall be set as follows:

- The K parameter shall be set to RK.
- The fi parameter shall be set to 0x45.
- The RAND parameter shall be set to TK_RAND|TK_RAND.
- The Fmk parameter shall be set to 0x41484147.

TK shall be set to the computed 128-bit output of *f3*.

## 4.5.4 SK

### 4.5.4.1 SK_RAND

SK_RAND shall be 32 bits in length.

SK_RAND shall be generated by invoking the algorithmic function *f0* as specified in Section 2.2.2.2 of S.S0055.

The input parameters of f0 shall be set as follows:

- The K parameter shall be set to a randomly chosen seed.
- The fi parameter shall be set to 0x41.
- The Fmk parameter shall be set to 0x41484147.

SK_RAND shall be set to the least significant 32 bits of the computed 64-bit output of *f0.*

### 4.5.4.2 SK Generation

SK shall be 128 bits in length.

SK shall be generated by invoking the algorithmic function *f3* as specified in Section 2.2.2.8 of S.S0055.

The input parameters of *f3* shall be set as follows:

- The K parameter shall be set to BAK.
- The fi parameter shall be set to 0x45.
- The RAND parameter shall be set to SK_RAND|SK_RAND|SK_RAND|SK_RAND.
- The Fmk parameter shall be set to 0x41484147.

SK shall be set to the computed 128-bit output of *f3*.

# 5 Motivation

## 5.1 Design Philosophy

The major threat addressed in this document is that of user(s) accessing the BCMCS content without paying the fees. This threat applies only when access is controlled on a subscription basis.

### 5.1.1 A Specific Goal

To access the BCMCS content, a user must have the current decryption keys. The UIM is not powerful enough to decrypt the content so the ME must perform the decryption. This implies that the decryption keys must be stored in the ME, which cannot be considered a secure storage device. It must be considered that eventually an attacker may find a way to extract the current decryption key from the ME. An attacker who is a subscribed user will then be able to distribute the decryption key to other non-subscribed users. In summary, the need to store decryption keys in unsecure memory makes it impossible to design a scheme where non-subscribed users CANNOT access the data.

We must recognize that the most we can do is dissuade the potential market (those users for which the service is targeted) from using illegitimate means to access the content.

Assuming the primary threat is subscribed users distributing decryption keys to non-subscribed users, the solution is for the decryption key to change frequently and in an unpredictable manner. The challenge is achieving this while minimizing the transmission overhead required for key distribution. The solution described herein is to distribute a Broadcast Access Key (BAK) to each user individually, and for many decryption keys to be derived using the BAK and public information sent with the BCMCS. The BAK is stored in the UIM.

## 5.2 Motivation for Establishing Registration Keys

Recall that the purpose of the BCMCS security is to prevent unauthorized (non-subscribed) users from accessing subscription-based content. Consequently, an unauthorized user should not be able to obtain BAK values. An unauthorized user may try to obtain BAK from a BAK Distributor (BAKD) by providing the identity of a subscribed UIM during a BAK request. If the unauthorized user can control what key is used to encrypt BAK, then the unauthorized user will be able to decrypt the packets sent back by the BAKD, and obtain BAK.

The best way to counter this attack is to encrypt the BAK using a key known only to the correct UIM (that is, the UIM identified in the BAK request). In this case, other UIM's will not be able to decrypt the packets sent back by the BAKD. This provides implicit authentication of the UIM.

*3GPP2 S.P0083*

*Version 0.6*

*Version Date: 5 June, 2003*


# Broadcast-Multicast Service Security Framework

***EDITOR***

Philip Hawkes
QUALCOMM Australia
+(61-2) 9817-4188
phawkes@qualcomm.com

***REVISION HISTORY***

| ***REVISION HISTORY*** | | |
|---|---|---|
| ***Rev. 0.1*** | *First draft* | *11 September 2002* |
| ***Rev. 0.2*** | *Second draft: Simplified description to make it less architecture-dependant.* | *28 October 2002* |
| ***Rev. 0.3*** | *Third draft: Merged Lucent and Qualcomm proposals. This version was approved to be the baseline text at conference call 2002/11/19.* | *19 November 2002* |
| ***Rev. 0.4*** | *Fourth draft: major changes and clarifications.* | *11 December 2002* |
| ***Rev 0.5*** | *Title changed from "Broadcast Security Framework" to "Broadcast-Multicast Service Security Framework".* | *13 January 2003* |
| ***Rev 0.6*** | *Terminology changes to reflect decisions in TSG-X BCMCS Ad Hoc. Also some minor editorial changes.* | *5 June 2003* |
| ***Rev 0.7*** | *BCMCS Security Algorithhms specified ( Section 4.5)* | *15 July 2003* |

# Table of Contents

# 1 Introduction and Scope

This document defines the security framework for the Broadcast-Multicast Services (BCMCS). The security framework provides a logical description of the security information, functions and protocols for BCMCS. The architectural design of the network that supports these functions is outside the scope of this document.

# 2 References

S.S0053

S.S0055

# 3 Definitions and Abbreviations

**BAK** 128-bit Broadcast Access Key: Provides access to one or more multicast IP flows of a particular set of BCMCS programs for a certain amount of time (for example, one day, week or month). Each encrypted set of BCMCS programs have a different BAK value. Each BAK should have the following associated values:

**BAK_ID:** BAK identifier. A sequence number that identifies which value of BAK is currently valid for a particular BCMCS Multicast IP Flow; that is, the BAK is identified by the combination of a BCMCS_FLOW_ID and BAK_ID. For a particular BAK, the corresponding value of BAK_ID is the same for all users.

**BAK_Expire:** Indicates when the BAK will expire. This may be indicated by the time when the BAK will expire, or by the Delta time left until the BAK expires, in which case it is calculated when a new BAK is received. Therefore, there may be multiple BAK_Expire values associated with the same BAK. Before the BAK_Expire time expires, the MS is expected to request a new BAK value.

**BCMC Content** Broadcast-Multicast Content, also referred to simply as **Content**. The content is the data being broadcast. Typically, this is expected to be audio-visual data.

**BCMCS** Broadcast-Multicast Service. BCMCS is the name of the system offering broadcast and multicast services.

**BCMCS Content Stream** A single BCMCS program identified by content name. This is also referred to simply as a **content stream.** A content stream can be considered as equivalent to a TV channel. For example, the document may refer to a “CNN BCMCS content stream” or “local news BCMCS content stream”. A BCMCS Content Stream may consist of multiple Multicast IP Flows.

**BCMCS_FLOW_ID** A value used for identification of a BCMCS Multicast IP Flow.

**CS** Content Source (functional entity): Provides the unencrypted data for the service. A CS may be a proxy or an aggregator for other content servers, but from a network point of view, it is the provider of content to the MS.

**BAKG** BAK Generator (functional entity): Generates BAK, determines BAK_ID and BAK_Expire, and delivers them to the BAKD and SKM. BAKs should be generated in a secure manner and appear random.

**BAKD** BAK Distributor (functional entity): Obtains authorization and TK from SM and encrypts BAK for delivery to UIMs.

**CE** Content Encryptor (functional entity): Encrypts content from the CS, and sends the encrypted content to the MS via the cellular system.

**MS** Mobile Station: For the purposes of this document, the MS is considered as two separate entities, the UIM and ME. Note that this does not preclude the use of a stand-alone secure MS (trustworthy mobile equipment with internal secure memory).

- **UIM**: User Identity Module (UIM): The UIM is a low power processor that contains secure memory. The UIM may be removable (like a SIM card) or part of the MS itself. The UIM calculates the SK used by the ME to decrypt the content.
- **ME**: Mobile Equipment: The ME contains a high power processor, but no secure memory. It uses the SK, received from the UIM, to decrypt the content.

**Multicast IP Flow**. Similar to an ordinary IP flow, with the exception that the destination address is an IP Multicast address. The flow can be identified by source address, source port, destination IP Multicast address, and destination port.

**RK** Registration Key: provisioned in the UIM and SM prior to service. Each SM should have a unique 128-bit RK for each UIM.

**SK** Short-term Key: The 128-bit SK is used by the CE to encrypt the BCMC content and the SK is used by the MS to decrypt the content. The SK is calculated in the SKM and the UIM. The SK is derived from SK_RAND and BAK.

**SK_RAND**: SK Random Number used to calculate a unique SK.

**SKM** SK Manager (functional entity): Generates SK using BAK and SK_RAND and passes SK, SK_RAND and BAK_ID to the CE.

**SP** Service Provider: a network providing voice/data services.

- **Visited SP** The serving network in which the MS is currently located.
- **Home SP** The network holding the user's voice/data subscription.

**SM** Subscription Manager (functional entity): performs accounting, authentication and authorization for BCMCS. The SM also calculates the TK, based on the RK, used to hide the BAK. The SM may be the subscriber's home AAA (H-AAA) or be an independent entity.

**TK** 128-bit Temporary Key: used by the BAKD to encrypt BAK when provisioning BAK in the UIM. The TK is obtained from the SM.

# 4 Overview of BCMCS

## 4.1 Introduction to BCMCS

The aim of BCMCS is to provide broadcast/multicast service to authorized (subscribed) users. The Content Sources (CSs) provide content, via a cellular serving system, to the BCMCS subscribers. The content may be IP multimedia messages such as audiovisual data or broadcast multimedia messages. A CS may be part of the serving network, or an independent entity. If the content is offered for free, then any user may view/process the content. If access to the BCMCS is subscription based, then the content can be encrypted so that only the authorized users can view/process the content. This document addresses the security needs for BCMCS.

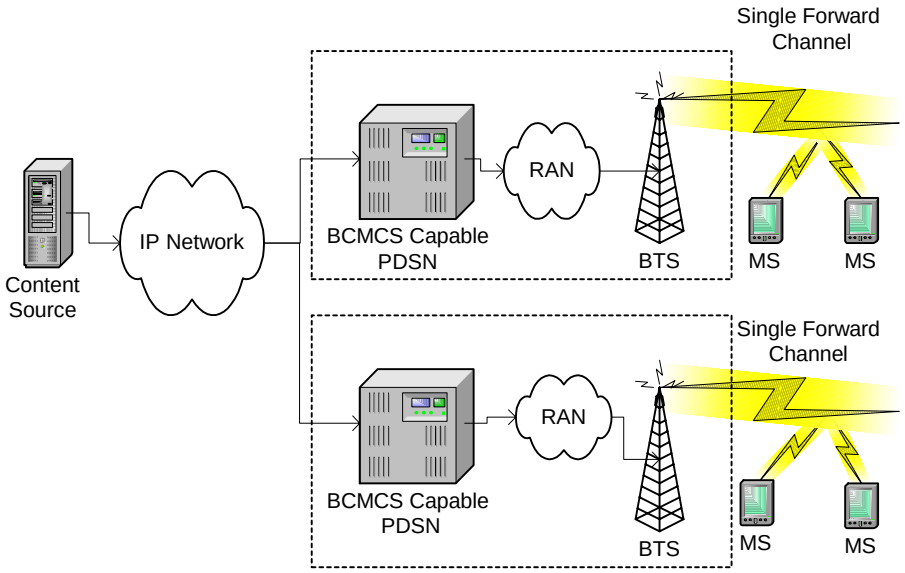


Figure 4.1 High Level view: Broadcast/Multicast Service

A single CS may provide more than one BCMCS to one or more carriers. The users may subscribe to services through their own carrier, or directly from a CS. The entity that supports the user subscription profile is the *Subscription Manager* (SM). The user may subscribe to more than one service through one or more SMs. The SMs provide service authorization and are an integral part of the BCMCS key management.

## 4.2 Summary of Key Management

For the purposes of this document, the Mobile Station (MS) is considered as two separate entities: the User Identity Module (UIM) and the Mobile Equipment (ME). The UIM is a low power processor that contains secure memory. The ME contains a high power processor, but no secure memory. The ME and the UIM may be integrated within one physical unit. In fact an MS can even be a trustworthy ME with its own internal secure memory.

The user authentication and authorization for cellular services, as well as network security is outside the scope of this document. This document assumes the following trust model: all network entities are presumed to trust the other network entities that they communicate with, all communication between network entities is presumed to be secure, and all network entities are presumed to perform their tasks correctly. The UIM is trusted by the SM and vice versa. All network entities trust the UIM to keep secrets and perform its tasks correctly. However, the UIM trusts only the SM.

The main threat addressed by this document is the threat of a user obtaining cheap and reliable access to the content of a particular content stream without being authorized for that particular content stream. To counter this threat, the content is delivered encrypted to the ME, and the decryption keys are provided only to those users who are authorized (subscribed) to receive

BCMCS. The primary focus of this document is the key management scheme for content streams that require authorization.

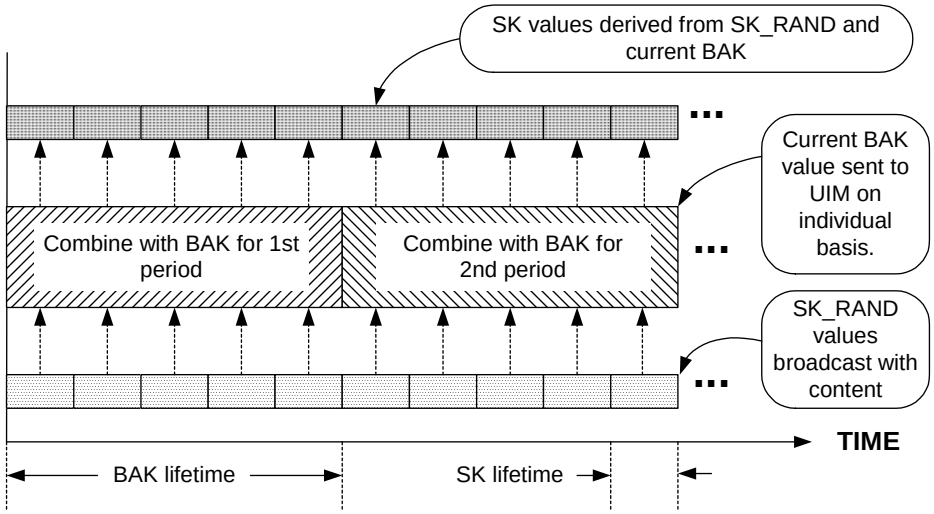


Figure 4.2. BCMCS Key Management. Multiple SK values are generated for each BAK value. The SK values are generated by combining BAK with the SK_RAND value that is transmitted with the encrypted content.

The content is encrypted using a unique and frequently changing *Short-term Key* (SK). The ME decrypts the content using the same SK. The SK should be frequently changed to minimizing the impact of a "rogue shell" sharing its SK with unauthorized users. The SK is never transmitted over the air; it is derived by the UIM from a *Broadcast Access Key* (BAK) and a random value broadcasted along with the encrypted content.

In order for the user to decrypt the necessary BAK values, the user's UIM must share an RK with an SM. Provisioning of RK is outside the scope of this document. A temporary encryption key (TK) is derived from the RK by the SM, sent to the BAKD, and subsequently used by the BAKD to encrypt/decrypt the BAK.

The BAK is encrypted by the BAK Distributor (BAKD), which may be associated with the Content Source, with the Subscription Manager, or the visited cellular system. The same BAK is provided to all users granted access to a particular Multicast IP Flow, and it is active for a pre-defined period of time. The multiple Multicast IP Flows associated with the particular BCMCS content stream may share the same BAK. Once the UIM has obtained the encrypted BAK (from the BAKD), it recovers the BAK (with its own calculated TK), and then computes the SK value needed to decrypt the Multicast IP Flow. The SK is then delivered to the ME for content decryption. The BAK is never divulged to the ME, and only the UIM with a correctly pre-provisioned RK can recover the BAK.

## 4.3 Security Functional Architecture

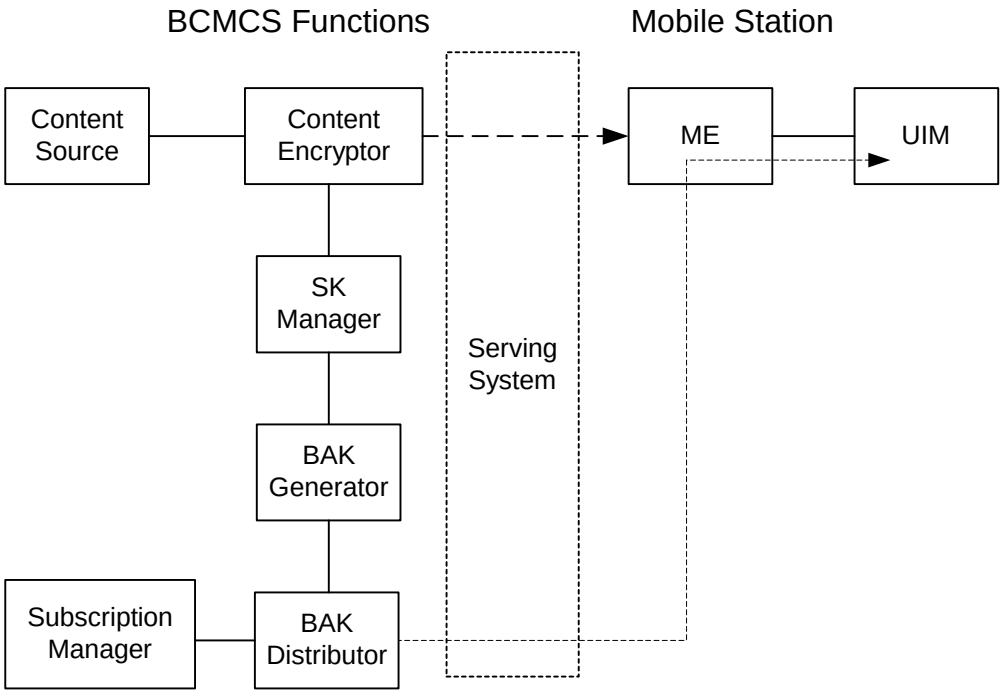


Figure 4.3. Functional Architecture

Figure 4.3 shows the functional entities that may be involved in BCMCS. Those entities represent functions essential to support secure BCMCS. They could be incorporated in one or more physical entities in the network. It should not be assumed that these functional entities correspond to separate architectural entities, so not all interfaces will require standardization. The allocation of the security functions to network entities is outside the scope of this document.

The BCMCS functional entities are described in more detail below.

### 4.3.1 Summary of Functional Entities

**CONTENT PATH**

**Content Source (CS)**: Generates the un-encrypted content.

**Content Encryptor (CE)**: Encrypts the content using the SK provided by the SKM. Encryption may take place at the Content Source or in the serving cellular system.

**KEY/AUTHORIZATION PATH**

**Subscription Manager (SM)**: May provide the functions of Authentication, Authorization and Accounting (AAA). The SM shares a Registration Key RK with the UIM: This key RK may be the A-key (which is the basis of key distribution and authentication for voice/data services as described in S.S0053), the key K used for AKA (as described in S.S0055), or some other key provisioned specifically for BCMCS. The SM calculates the TK, based on the user specific RK. *The provisioning of RK in the UIM and SM is beyond the scope of this document.*

**BAK Generator (BAKG):** Generates the Broadcast Access Key (BAK) as required, and distributes BAK and to BAKD and SKM.

**BAK Distributor (BAKD):** Controls the distribution of the Broadcast Access Key (BAK).

**SK Manager (SKM):** Controls the updating and distribution of the Short-term Key (SK).

**Mobile Station (MS)** For the purposes of this document, the MS is considered as two separate entities, the UIM and ME.

**UIM**: User Identity Module : The UIM shares a key RK with the SM. The UIM performs all key management related to BCMCS.

**ME**: Mobile Equipment. Includes equipment for receiving the BCMCS. The ME performs decryption of encrypted content using SK obtained from the UIM.

### 4.3.2 Summary of Key Distribution

Figure 4.4 shows the basic communications involved in the key distribution for BCMCS encryption. Many details are omitted in this diagram for the sake of clarity. The Figure is described below.

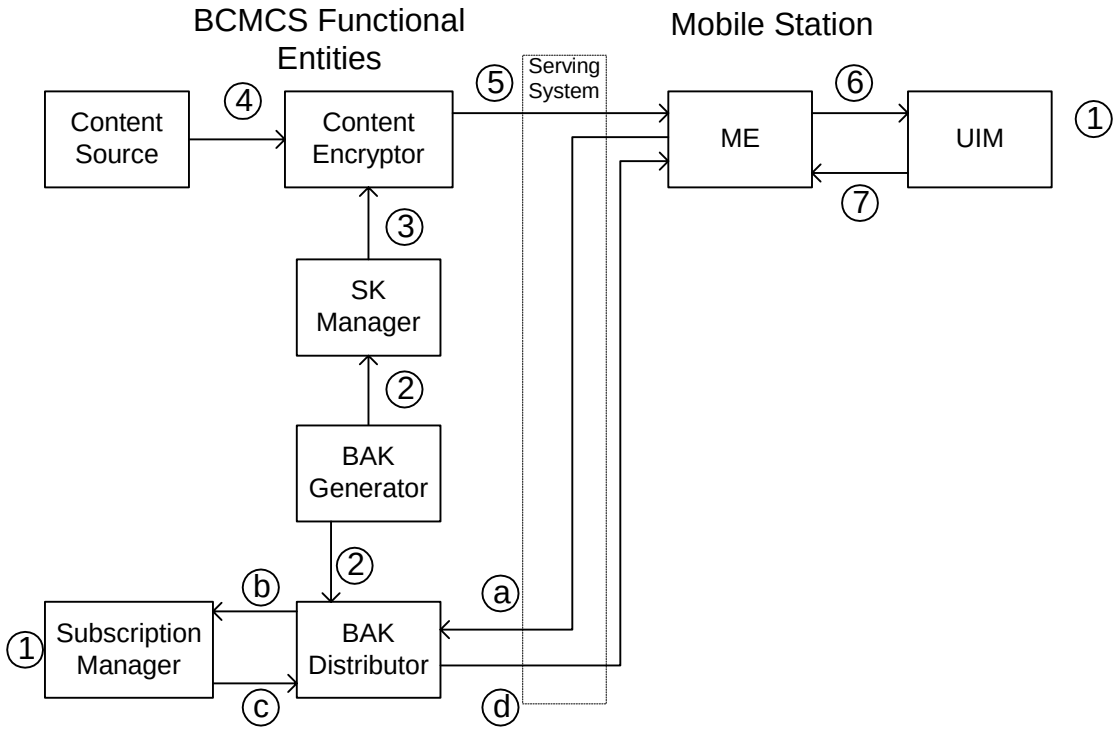


Figure 4.4 Logical Architecture showing functional entities involved in BCMCS security.

#### 4.3.2.1 BAK Generation

The following steps (1-2) are performed prior to the transmission of encrypted Multicast IP Flow(s) to the MS:

1. The UIM and SM are provisioned with a Registration Key RK that will be the basis of authentication and key exchange with respect to BCMCS.

2. The BAKG generates a value for BAK, and associates the value with an identifier BAK_ID and an expiry time (BAK_Expire). The value of BAK, along with the corresponding values of BAK_ID and BAK_Expire are passed to the SKM and BAKD.

#### 4.3.2.2 Normal Procedure for Encrpting a Multicast IP Flow With Unchanged BAK

The following steps (3-7) occur under normal circumstances, when BAK is unchanged from the previous decryption operation in the MS.

3. The SKM creates SK from the current BAK and a random value SK_RAND. The SKM passes SK, SK_RAND, BAK_ID and BAK_Expire to the CE.

4. The CS sends Multicast IP Flow to the CE.

5. The CE encrypts the Multicast IP Flow using SK and sends the encrypted Multicast IP Flow to the MS via the serving system. The CE also includes SK_RAND and BAK_ID with the encrypted Multicast IP Flow.

6. The ME receives the encrypted Multicast IP Flow, and takes action as follows:

   a. If BAK_ID and SK_RAND are unchanged from the last received Multicast IP Flow, the ME decrypts the Multicast IP Flow using the value of SK currently assigned to that Multicast IP Flow and passes the result to the user application;

   b. If BAK_ID or SK_RAND have changed, the ME requests a new SK from the UIM, including the BCMCS_FLOW_ID, BAK_ID and SK_RAND.

7. The UIM generates SK from BAK and SK_RAND and returns SK to the ME, which decrypts the Multicast IP Flows and passes the result to the user application.

#### 4.3.2.3 Procedure With Changed or Unavailable BAK

Steps (a-d) in Figure 4.4 are performed to request a new BAK. Section 4.4 discusses possible methods by which the MS determines that a new BAK is needed.

a. The ME sends a BAK request to the BAKD. . The BAK request may include authentication information based on RK, which can be used by the SM to determine that the request came from a legitimate subscriber. Section 4.4.1 discusses the need for authentication of BAK requests.

b. In order to send BAK to the UIM, the BAK must be encrypted to protect it against reception by other than the intended recipient. The BAKD requests a temporary key (TK) generated by the SM.

c. The SM generates TK from a random value TK_RAND and RK. TK_RAND may be generated by the BAKD, or by the SM. TK_RAND may also be used as a challenge in the authentication process described in step a. The SM sends TK and TK_RAND to the BAKD.

d. The BAKD encrypts the value of BAK with TK, and sends the encrypted BAK, along with TK_RAND and BAK_Expire to the UIM via the ME. The UIM first forms TK from TK_RAND and RK, and then decrypts the encrypted BAK with TK to form BAK. The value of BAK and its associated BAK_Expire are stored in the UIM. The UIM should be able to store at least two values of BAK so that a new BAK can be obtained and stored in anticipation of the BAK expiration time.

## 4.4 BAK Management

A user can repeat the BAK update process with multiple content streams. Thereafter (from a security perspective) the MS has the potential to be managing the decryption keys for multiple content streams simultaneously. In order to determine which BAK corresponds to which Multicast IP Flow, a BCMCS flow identifier BCMCS_FLOW_ID should be stored with the BAK.

This document does not specify how an MS determines that it needs to update BAK. We assume that a means will be provided for the MS to determine that its BAK is about to expire or has expired, triggering action to perform a BAK update. Several methods are possible for accomplishing this. The specific method used is not important for the present subject.

**BAK_ID**. The UIM can determine if a BAK is associated with a particular Multicast IP Flow by referring to the corresponding BCMCS_FLOW_ID. However, there will be times when the UIM needs to store two BAK values associated with a particular BCMCS_FLOW_ID. The reason is that the BAK must be updated in the UIM prior to the BAK actually being used to encrypt/generate SK values (see Section 4.4.3). Consequently, both the old BAK and the new BAK will reside in the UIM simultaneously. The UIM must have some means to determine which BAK is valid. The recommended method is to allocate a BAK identifier BAK_ID to each BAK, in addition to the BCMCS_FLOW_ID. The BAK is identified by the BCMCS_FLOW_ID and BAK_ID. The CE will include the BAK_ID with the encrypted content in the broadcast data. The UIM can now distinguish if the old BAK is still being used or if the new BAK has begun to be used.

A value for BAK_Expire will be provided along with the BAK, so the MS can update expired keys.

**BAK lifetime.** The BAKG decides how often BAK is changed. The following issues should be considered in deciding how often BAK is to be changed.

- Frequent BAK changes will provide more security.

- Frequent BAK changes will also provide greater flexibility in subscription control. We show this by example. Once a user has BAK, they can access the content for the lifetime of that BAK. Suppose the BAK is changed at the beginning of every month. If a user's subscription runs out halfway through the lifetime of a BAK, the user will still be able to generate SK (and thus view the content) until the BAK expires. So by changing BAK only every month, the SM can only charge subscriptions from the beginning of the month to the end of the month. A user can't subscribe from the middle of one month to the middle of the next. However, if BAK changed every day, then the user could subscribe from the beginning of any day during the month.

- Increasing the frequency of BAK changes should be evaluated against a possible increase in the number of times the mobile station has to retrieve new BAK values.

### 4.4.1 Authenticating BAK Requests

An adversary cannot obtain BAK by performing a BAK request while impersonating a subscribed user. Only the subscribed user will be able to derive TK from RAND_TK, and thus extract BAK. For this reason, the BAKD does not need to authenticate BAK requests in order to protect BAK.

If, however, other control functions or accounting data are included with the BAK request, then it is necessary to prove the authenticity of the data source. It may also be desirable to authenticate requests to prevent malicious generation of excess network traffic. *The procedures for authenticating the request are for further study.*

Note that even though authentication of BAK requests may not be needed, there is still a need for authorization of the request. Such authorization must be specific to the BCMCS to which the requested BAK applies.

### 4.4.2 Storage of BAK

It is essential to the security model used herein that the UIM does not reveal BAK. If a single UIM reveals BAK, then all security is compromised until the BAKG changes BAK.

The UIM should store BAK and related data about BAK, such as BAK_ID and expiration time, if any. The BAK is identified by the BCMCS_FLOW_ID (of the corresponding BCMCS Multicast IP Flow) and BAK_ID.

The ME may store the BAK-related data, to save requesting this information from the UIM.

### 4.4.3 Updating BAK before use

It may prove beneficial to provision UIM with BAK shortly before BAK begins being used to derive SK values. Otherwise, once the CE starts sending packets with SK derived from the new BAK, the user would experience a delay as the MS performs a BAK update. If many users are tuned in, then there will be a burst of traffic as all the MSs perform a BAK update.

To avoid such problems, the BAKD should allow an MS to obtain the new BAK shortly before the BAK changes. Different MS may have different schedules for performing BAK updates, to prevent too many MSs performing a BAK update at once.

For security reasons, BAK should be distributed as close as possible to the time of use.

## 4.5 BCMCS Security Algorithms

### 4.5.1 Encryption of Content

[Editor's note: We prefer that the encryption algorithm be based on AES. However the exact mechanism depends on the protocol layer at which the encryption is performed, and this has not yet been decided.]

#### 4.5.1.1 Content Encryption Using Upper Layer protocol

The content of BCMCS shall be encrypted by the upper layer protocols as defined by IETF, protocols capable of delivering BCMCS content. One such protocol is described in the SRTP draft (draft-ietf-ave-srtp-08). The stream encryption key is the Session Key (SK) defined below.

#### 4.5.1.2 Content Encryption Using Link Layer Encryption

TBD.

### 4.5.2 Encryption of BAK

BAK encryption (under key TK) shall be performed using the ESP_AES algorithm as defined in S.S0055.

The input parameter of ESP_AES shall be set as follows:

- The encryption key parameter shall be set to TK.
- The *fresh* parameter shall be set to TK_RAND.
- The *freshsize* parameter shall be set to 8.
- The *buf* parameter shall be set to the address of the octet containing the first bit of the buffer that contains the data to be encrypted or decrypted.
- The *bit_offset* parameter shall be set 0.
- The *bit_count* parameter shall be set to 128.

### 4.5.3 Management of TK

#### 4.5.3.1 TK_RAND

TK_RAND shall be 64 bits in length.

~~TK_RAND shall be generated in a manner that minimizes the probability that the same TK_RAND is used to create a TK for different BAK encryptions destined to the same terminal. The probability should be the same as for random selection of TK_RAND.~~

TK_RAND shall be generated by invoking the algorithmic function *f0* as specified in Section 2.2.2.2 of S.S0055.

The input parameters of f0 shall be set as follows:

- The K parameter shall be set to RK.
- The fi parameter shall be set to 0x41.
- The Fmk parameter shall be set to 0x41484147.

TK_RAND shall be set to the computed 64-bit output of *f0*.

#### 4.5.3.2 TK Generation

TK shall be 128 bits in length.

~~TK shall be generated from TK_RAND and RK using a secure one-way function chosen to minimize the likelihood that an adversary can obtain TK with knowledge of TK_RAND alone, and to minimize the likelihood that an adversary can obtain RK with knowledge of TK_RAND and TK. A standardized SHA-1 based function will be used for this purpose.~~

TK shall be generated by invoking the algorithmic function *f3* as specified in Section 2.2.2.8 of S.S0055.

The input parameters of *f3* shall be set as follows:

- The K parameter shall be set to RK.
- The fi parameter shall be set to 0x45.
- The RAND parameter shall be set to TK_RAND|TK_RAND.
- The Fmk parameter shall be set to 0x41484147.

TK shall be set to the computed 128-bit output of *f3*.

## 4.5.4 SK

### 4.5.4.1 SK_RAND

SK_RAND shall be 32 bits in length.

~~SK_RAND shall be generated in a manner that minimizes the probability that the same SK_RAND is used to create an SK for different content encryptions destined to the same terminal. The probability should be the same as for random selection of SK_RAND.~~

~~SK_RAND shall be generated in a manner that minimizes the probability that an adversary can guess the next value of SK_RAND with knowledge of previous values. This requirement makes it more difficult for a modified ME shell to compute future SK values and distribute the values to other MEs.~~

SK_RAND shall be generated by invoking the algorithmic function *f0* as specified in Section 2.2.2.2 of S.S0055.

The input parameters of f0 shall be set as follows:

- The K parameter shall be set to a randomly chosen seed.
- The fi parameter shall be set to 0x41.
- The Fmk parameter shall be set to 0x41484147.

SK_RAND shall be set to the least significant 32 bits of the computed 64-bit output of *f0*.

#### 4.5.4.2 SK Generation

SK shall be 128 bits in length.

~~SK shall be generated from SK_RAND and BAK using a secure one-way function chosen to minimize the likelihood that an adversary can obtain SK with knowledge of SK_RAND alone, and to minimize the likelihood that an adversary can obtain BAK with knowledge of SK_RAND and SK.~~

SK shall be generated by invoking the algorithmic function *f3* as specified in Section 2.2.2.8 of S.S0055.

The input parameters of *f3* shall be set as follows:

- The K parameter shall be set to BAK.
- The fi parameter shall be set to 0x45.
- The RAND parameter shall be set to SK_RAND|SK_RAND|SK_RAND|SK_RAND.
- The Fmk parameter shall be set to 0x41484147.

SK shall be set to the computed 128-bit output of *f3*.

# 5 Motivation

## 5.1 Design Philosophy

The major threat addressed in this document is that of user(s) accessing the BCMCS content without paying the fees. This threat applies only when access is controlled on a subscription basis.

### 5.1.1 A Specific Goal

To access the BCMCS content, a user must have the current decryption keys. The UIM is not powerful enough to decrypt the content so the ME must perform the decryption. This implies that the decryption keys must be stored in the ME, which cannot be considered a secure storage device. It must be considered that eventually an attacker may find a way to extract the current decryption key from the ME. An attacker who is a subscribed user will then be able to distribute the decryption key to other non-subscribed users. In summary, the need to store decryption keys in unsecure memory makes it impossible to design a scheme where non-subscribed users CANNOT access the data.

We must recognize that the most we can do is dissuade the potential market (those users for which the service is targeted) from using illegitimate means to access the content.

Assuming the primary threat is subscribed users distributing decryption keys to non-subscribed users, the solution is for the decryption key to change frequently and in an unpredictable

manner. The challenge is achieving this while minimizing the transmission overhead required for key distribution. The solution described herein is to distribute a Broadcast Access Key (BAK) to each user individually, and for many decryption keys to be derived using the BAK and public information sent with the BCMCS. The BAK is stored in the UIM.

## 5.2 Motivation for Establishing Registration Keys

Recall that the purpose of the BCMCS security is to prevent unauthorized (non-subscribed) users from accessing subscription-based content. Consequently, an unauthorized user should not be able to obtain BAK values. An unauthorized user may try to obtain BAK from a BAK Distributor (BAKD) by providing the identity of a subscribed UIM during a BAK request. If the unauthorized user can control what key is used to encrypt BAK, then the unauthorized user will be able to decrypt the packets sent back by the BAKD, and obtain BAK.

The best way to counter this attack is to encrypt the BAK using a key known only to the correct UIM (that is, the UIM identified in the BAK request). In this case, other UIM's will not be able to decrypt the packets sent back by the BAKD. This provides implicit authentication of the UIM.