QUALCOMM

S3-030451

BCMCS Security Framework

Greg Rose (Australia)
Jim Semple (USA)
QUALCOMM
{ggr,c_jsempl}@qualcomm.com

Outline

- Security Goals
- BCMCS Key Hierarchy
- BCMCS Functional Architecture
- Security Mechanisms
  - Key Management
  - Encryption layer
  - BAK Update

Security Goals

- **The problem**
  - **UIM is not powerful enough to decrypt so ME must decrypt. Decryption keys must be stored in the ME**
  - **ME is not secure storage. Must assume an attacker may extract the current decryption key from the ME**
  - **An attacker who is a subscribed user will be able to distribute the decryption key to other non-subscribed users**
- **In summary:**
  - **The need to store decryption keys in insecure memory makes it impossible to design a scheme where non-subscribed users CANNOT access the data**

Security Goals

- **The goal of the security:**

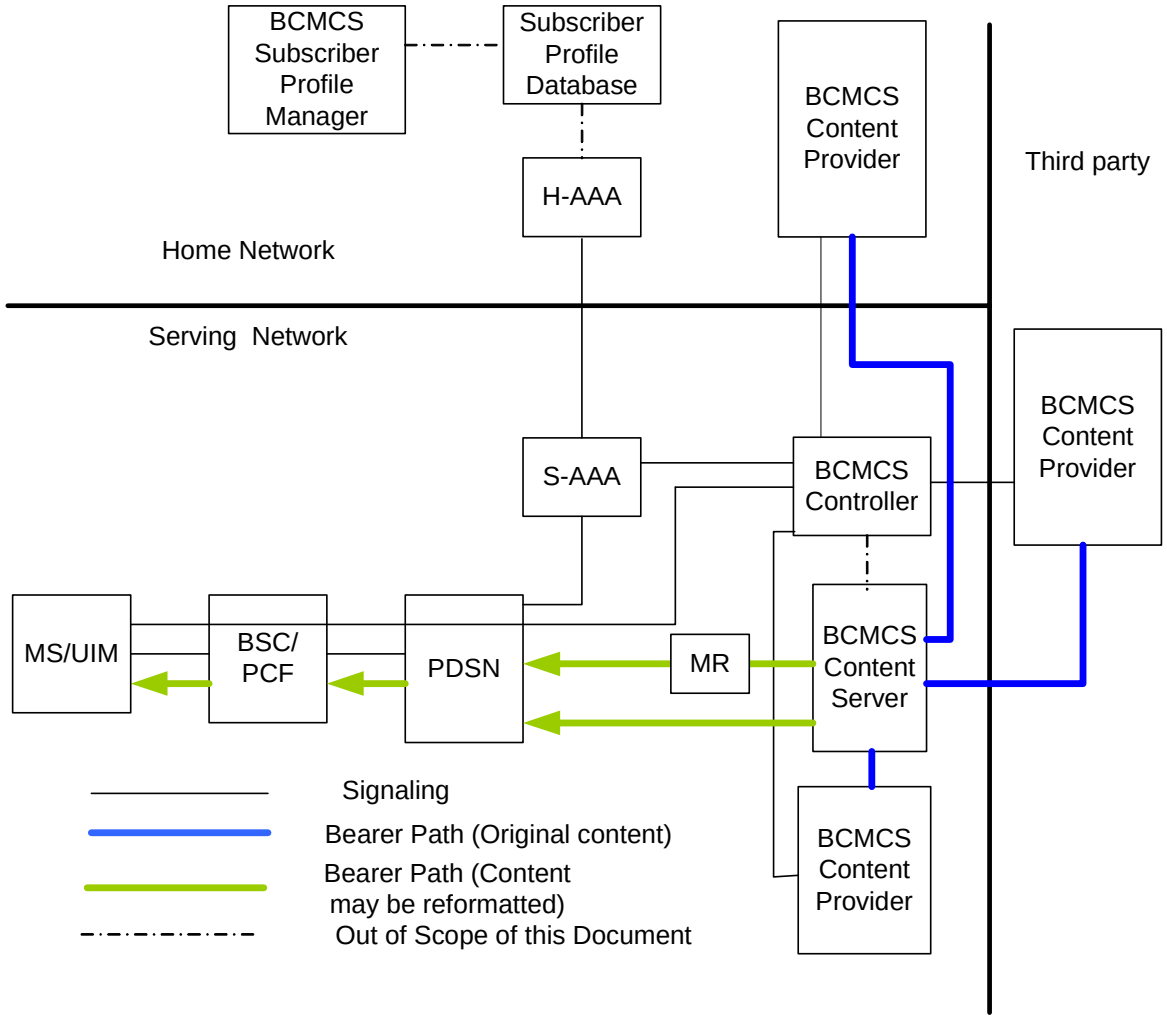
  Dissuade our potential market from using illegitimate means to access the content

- **What is the potential market?**
  - **Users that desire cheap access to multicast services while being mobile.**

- Attacks we should not be concerned about:
  - Attacks that are expensive to mount (per-user basis) and/or
  - Attacks that assume the user is not mobile.
  - E.g., we should not be concerned about attacks that require the user to perform a frequent data download of keys as mobile data downloads will be more expensive than BCMCS subscriptions.
- **This mechanism does not address integrity protection:**

BCMCS Architecture

(figure is from XP0019 v0.1.2)



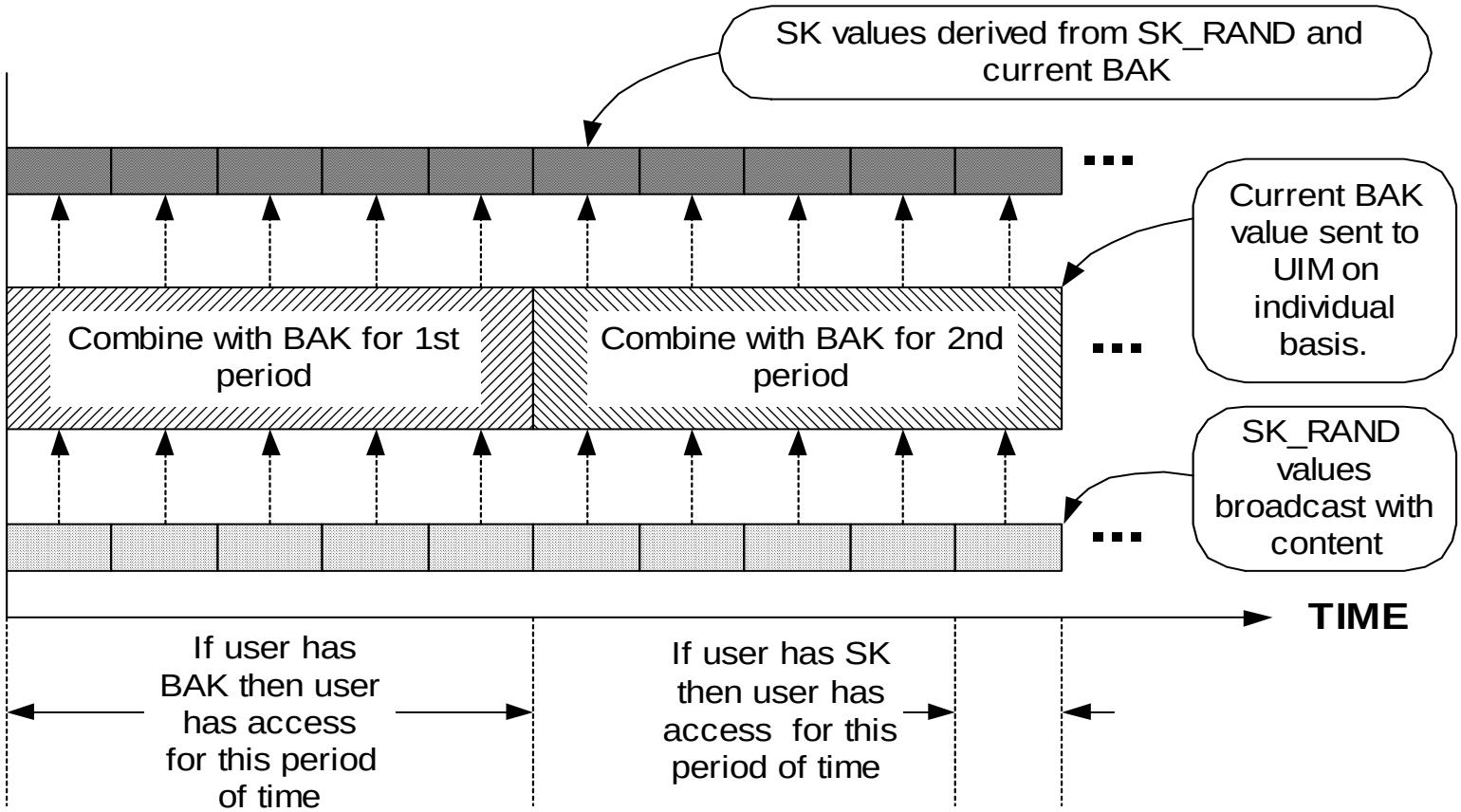
Some BCMCS Architecture Entities

- **BCMCS Controller:**
  - **responsible for managing and providing BCMCS session information to PDSN, MS, and the Content Server.**
- **BCMCS Content Server**
  - **BCMCS-CS is always in the Serving Network.**
  - **Makes BCMCS content available within an IP Multicast stream.**
  - **Not necessarily the creator or source of the content; it is the last application level entity to manipulate the content prior to content reaching PDSN.**
  - **If higher layer encryption enabled, the BCMCS-CS may encrypt the stream content. In this case, also serves function of SK Generator. May also serve the function of BAK Generator.**
- **BCMCS Content Provider**
  - **may be located in serving or home network or anywhere in an IP network (such as the Internet).**

BCMCS Key Hierarchy

- **BAK (Broadcast Access Key)** -medium term, shared key
  - **Multiple short-term keys (SK) derived from single BAK**
  - **Distributed to UIM of subscribed users on a per-user basis**
- **SK (Short-term Key)** – frequently changing, shared key
  - ***used to encrypt / decrypt content***
  - **Generated using SK_RAND which is sent in the clear with the encrypted content and BAK**
  - **UIM re-generates SK from BAK and SK_RAND, passes SK to ME**
  - **Changes frequently**
- **Keys used for Distributing BAK**
  - **RK (Registration Key)** – permanent, user-specific key
    - **Used to generate TK values / authenticate UIM**
  - **TK (Temporary Key)** – single use, user-specific key
    - **Used to encrypt BAK values, used by UIM to decrypt BAK values**
    - **Generated using RK**

Figure: BAK and SK



Using BAK and SK

- To encrypt (in network)
  - SK is generated from BAK and SK_RAND
  - (Multiple SK values generated from each BAK)
  - Content is encrypted with SK
  - Encrypted content is then transmitted along with SK_RAND
- To decrypt (in MS)
  - BAK sent to the UIM of authorized users on a per-user basis
  - ME receives encrypted content and SK_RAND
  - UIM generates SK from BAK and SK_RAND, and passes SK to the ME
  - ME decrypts content using SK

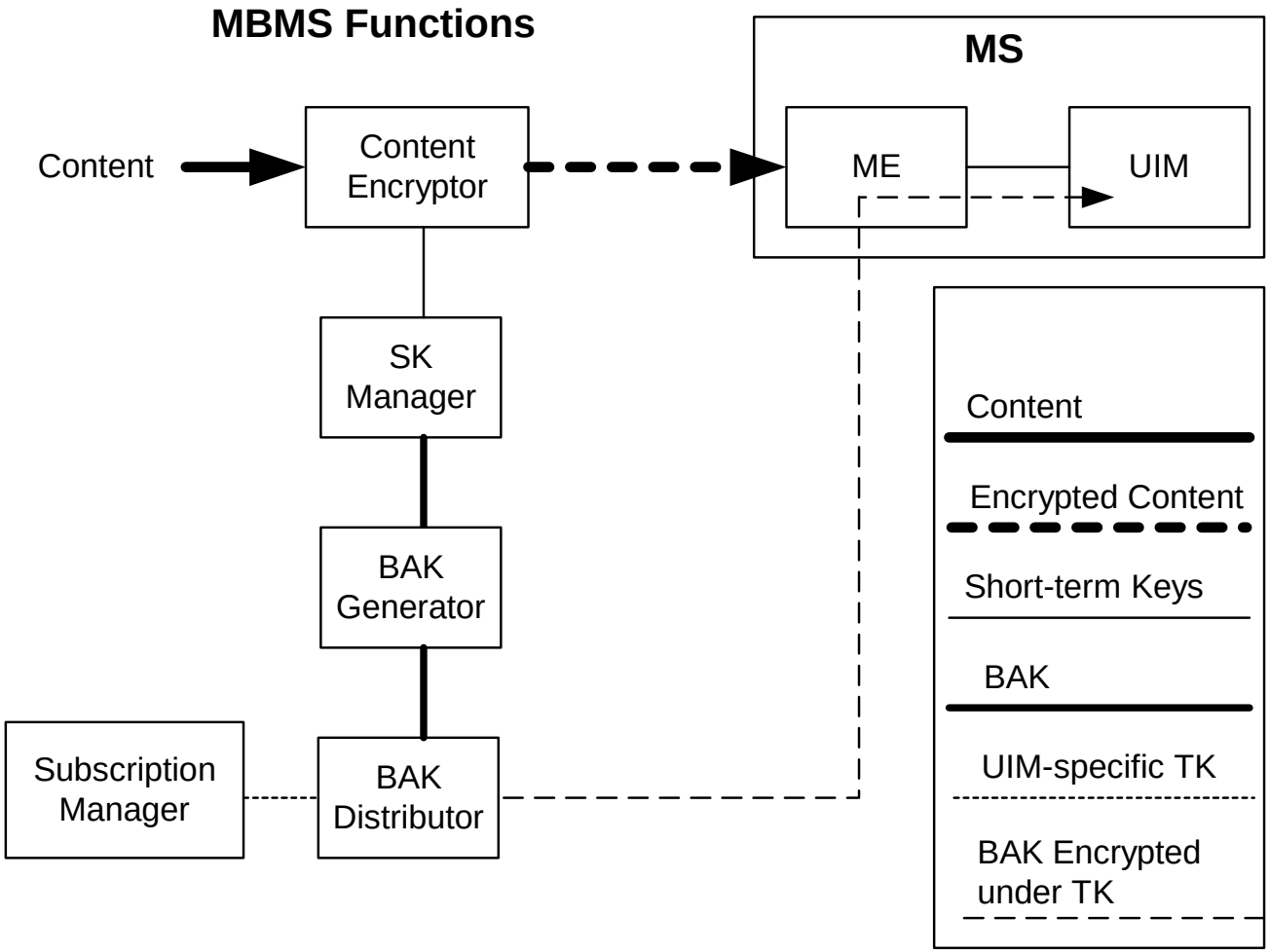
BAK and Subscriptions

- Once the user has a BAK value, the user can access ALL content that is encrypted while BAK value is being used to generate SK values
- Subscriptions grant users the right to be sent BAK values
  - Flat-rate subscriptions: the user is sent all BAK values over a certain time period
  - Event-based subscriptions: the user is sent all BAK values required to view a specific event
  - Usage-based subscriptions: the user is billed based on the number of BAK values used
- The time between changing BAK is a billing decision.
  - Time between BAK changes may vary for each multicast service
  - For a particular encrypted BCMCS service, the time between BAK changes need not stay constant

SK

- ME knows SK
  - Assume subscriber may distribute SK to other users so they can get free access to content
- By changing SK frequently, we limit the amount of time/data that SK is useful to other users
  - To access content, non-subscribed users must download SK values frequently. As discussed in security goals, this attack is not of concern
- The frequency of SK changes may vary for each encrypted BCMCS service and may vary with time
  - When determining how often to change SK, ensure that the cost of user downloading SK exceeds the value of content encrypted under SK

BCMCS Functional Architecture



Functional Entities

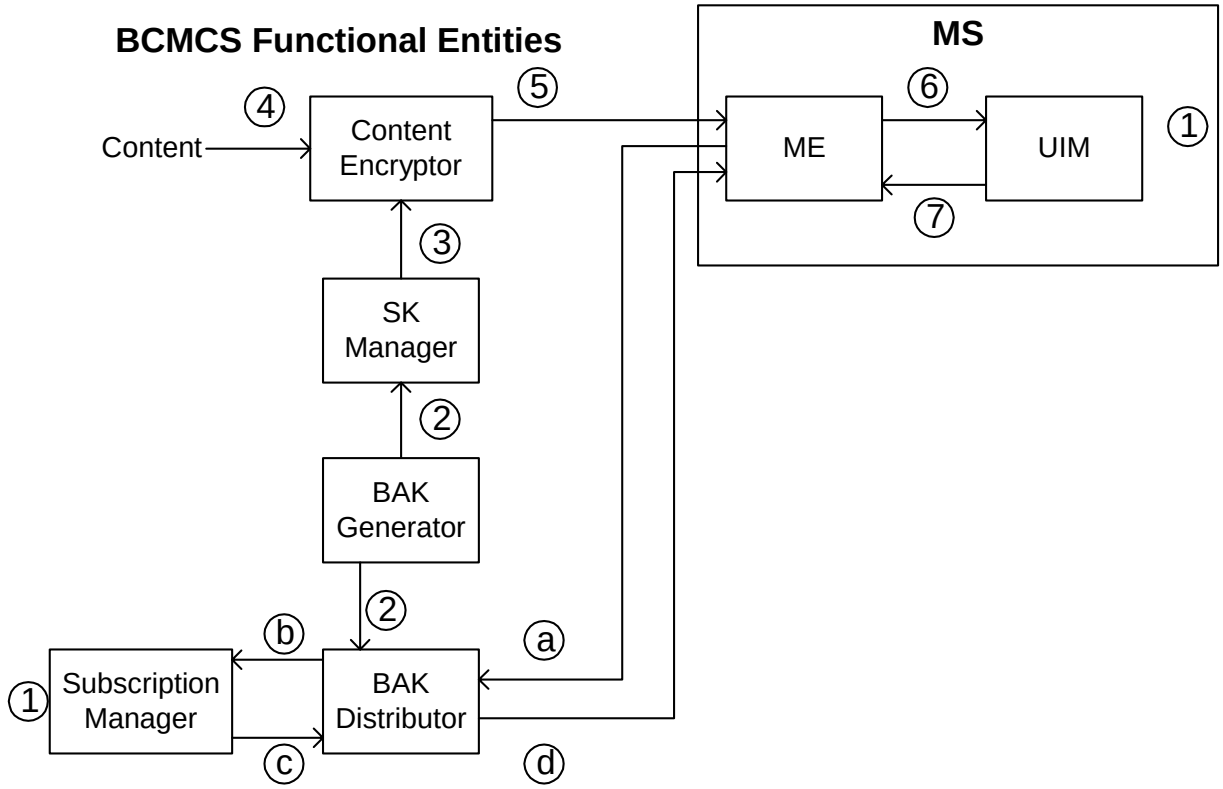
- **BCMCS Functions**
  - **Subscription Manager (SM):** holds subscription data authorizing user to some BCMCS services
  - **BAK Generator:** generates BAK: forwards to BAKD and SKM
  - **BAK Distributor:** encrypts BAK for provisioning into UIM
  - **SK Manager (SKM):** generates SK values from BAK and forwards to Content Encryptor (CE).
  - **Content Encryptor (CE):**
- **MS**
  - **Mobile Equipment (ME):** performs content decryption
  - **UIM:** performs key management

Keys and Functional Entities

- **RK (Registration Key)**
  - Used by SM to generate TK values / authenticate UIM
  - Held in Subscription Manager (SM) and UIM
- **TK (Temporary Key)**
  - Used by BAKD to encrypt BAK values, used by UIM to decrypt BAK values
  - Generated by SM using RK
- **BAK (Broadcast Access Key)**
  - Multiple decryption keys (SK) derived from single BAK
  - Generated by BAKG and forwarded to BAKD and SKM
  - Distributed by BAKD to UIM of subscribed users (on request)
- **SK (Short-term Key)**
  - Generated by SKM using BAK and forwarded to CE for encrypting content
  - Derived in UIM using BAK and passed to ME for decrypting content
  - Changes frequently

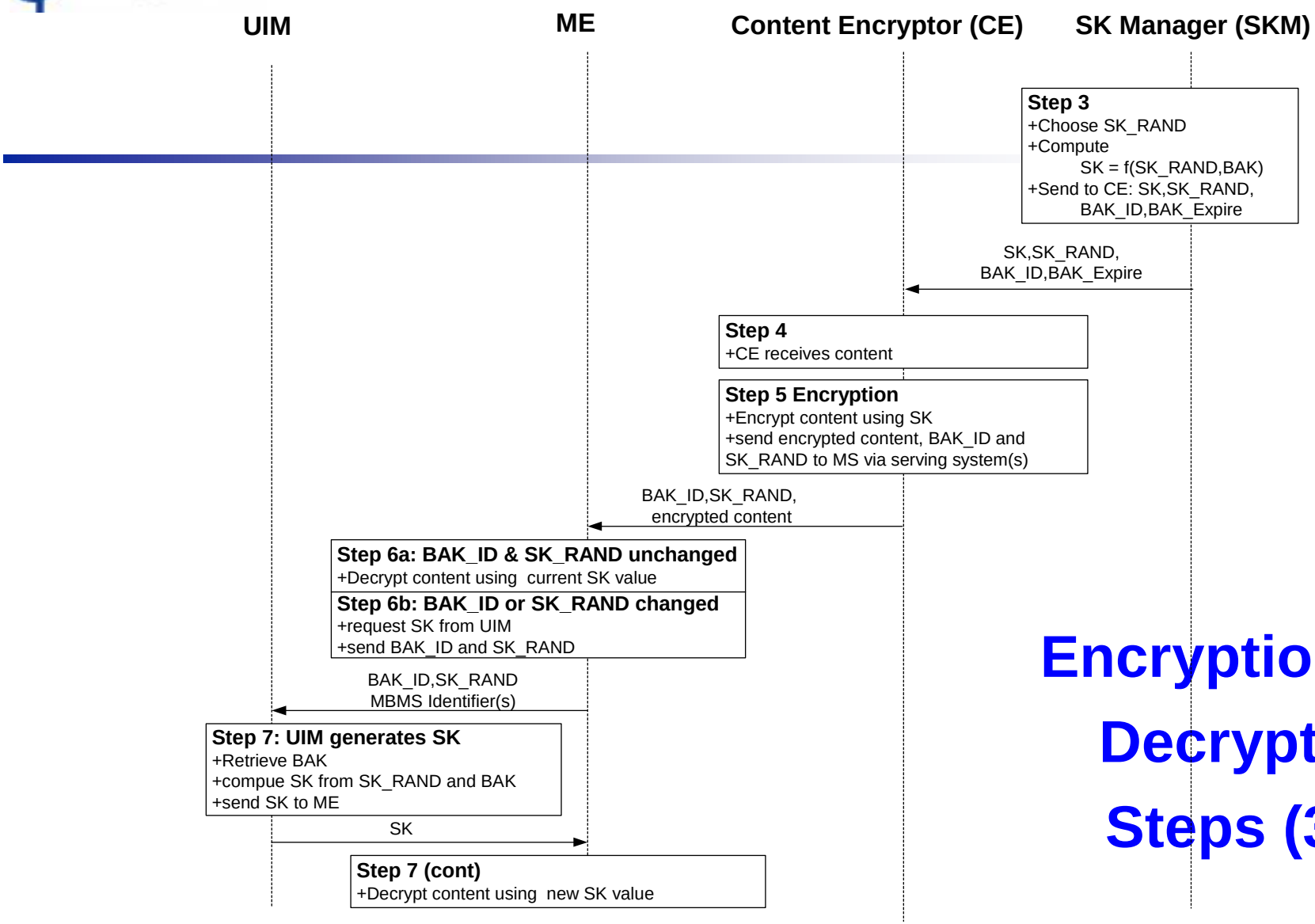
Security Mechanisms

- **Steps (1-2): Prior to ME decrypting Content**
- **Steps (3-7): Encrypting Content**
- **Steps (a-d): BAK Request**



Steps (1-2)

- **Step 1. UIM and SM provisioned with RK**
  - **Beyond scope of this document**
    - **Could be pre-provisioned in UIM before given to user**
    - **Could be provisioned OTA**
  - **This step only needs to occur once to associate SM and UIM**
- **Step 2. The BAKG generates a value for BAK**
  - **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**
  - **This step occurs only when the BAKG decides that a new value of BAK should be used.**



Encryption & Decryption Steps (3-7)

Encryption

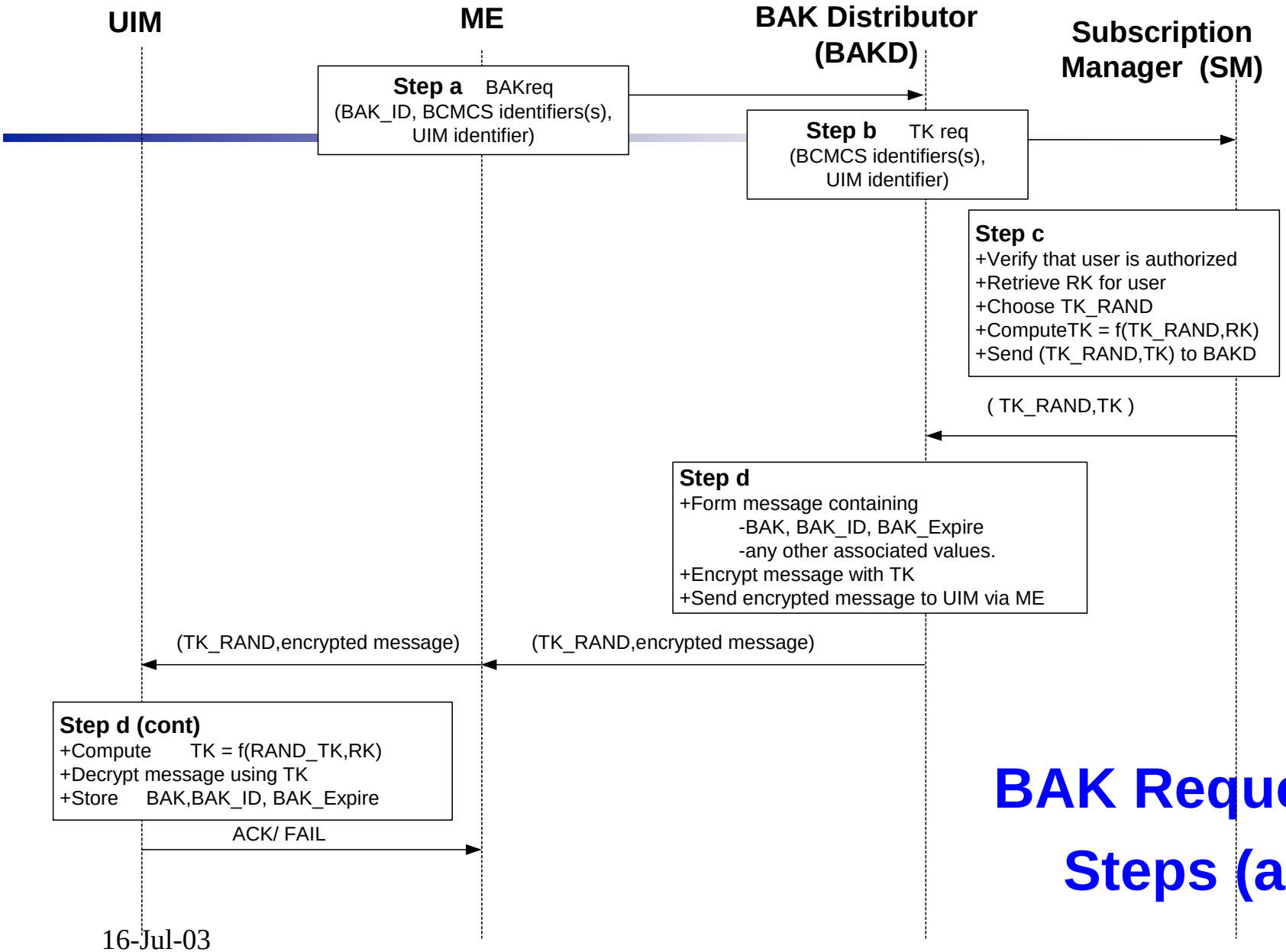
- **Step 3. SK Generation**
  - **The SKM creates SK from the current BAK and a random value SK_RAND. The SKM passes SK, SK_RAND, BAK_ID and BAK_Expire to the CE.**
  - **This step occurs frequently. Frequency may vary. The SKM should be instructed how often a new SK should be generated.**

Encryption of Content

- **Step 4. CE receives content**
  - **This step may be a continuous process**
- **Step 5. Encryption**
  - **CE encrypts content using SK and sends the encrypted content to the MS via the serving system(s). The CE also includes SK_RAND and BAK_ID with the encrypted content.**
  - **This step may be a continuous process. The CE uses a new SK when instructed by SKM**

Decryption of Content

- **Step 6.**
  - **a. If BAK_ID and SK_RAND are unchanged from the last received content, the ME decrypts the content using SK currently assigned to that content stream 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 BAK_ID and SK_RAND.**
  - **This step may be a continuous process.**
- **Step 7. UIM Generates SK**
  - **UIM generates SK from BAK and SK_RAND, and returns SK to the ME, which decrypts the content and (pending on step 6a) passes the result to the user application.**
  - **Usually occurs only when SK changes**



BAK Request Steps (a-d)

BAK Request

- **Occurs when the MS determines that a new BAK is required**
- **Step a. The ME sends BAK request to BAKD**
  - **Includes the BAK_ID of the BAK requested.**
  - **May include authentication information based on RK**
- **Step b. BAKD requests a TK from the SM**
- **Step c. SM Generates TK**
  - **If user is authorized to access this BCMCS service, then 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.**

BAK Request: cont.

- **Step d. The BAKD encrypts BAK with TK, and sends to UIM via ME**
  - **Includes TK_RAND, and other associated values BAK_ID and BAK_Expire.**
  - **UIM first forms TK from TK_RAND and RK…**
  - **.. and then decrypts the encrypted BAK with TK to form BAK etc.**
  - **The value of BAK and its associated values are stored in the UIM**

Comments on BAK

- BAKG controls how often BAK value changes
  - If BAKG is controlled by network, then the network operator controls how often BAK value changes
  - Regular changes are preferable since BAK_Expire can be more accurate
  - If desired, BAK can be changed in ad hoc manner: e.g., when users leave a user group