3GPP TSG SA WG3 Security — S3#29 **S3-030411**

15 – 18 July 2003

San Francisco, USA

Source: **Siemens**

Title: **Key provision in bootstrapping of application security**

Document for: **Discussion and decision**

Agenda Item: **7.9 subscriber certificates**

Abstract

This contribution proposes two improvements to the currently specified provision of the keys shared between a UE and a NAF, namely a reduced involvement of the HSS and a means for the NAF to request a key change. The changes are presented in the form of pseudo-CRs.

1. Problem statement

1.1 Establishment of key shared between UE and NAF

The draft TS “Bootstrapping of application security using AKA and support for Subscriber Certificates” v020 (S3-030317, May 2003) describes in section 4.3.1 how a UE and an application server (NAF) can obtain a shared key in a generic way from a run of a protocol A (likely to be http digest aka) between the UE and a Bootstrapping Server Function (BSF). The procedures using bootstrapped security association are described in section 4.3.2 of the TS. According to section 4.3.2, a user first has to run protocol A with the BSF in case he does not yet share a key with the NAF he is about to access. This has the following consequences:

1. **Heavy HSS involvement**: for each new NAF the user wants to access there is a new run of protocols A and C. This means that for each such case, the HSS is involved;
2. **lack of key separation**: a key agreed between the UE and the BSF using protocol A could be retrieved by any NAF if it presents the right transaction identifier to the BSF. As a consequence, the UE does not know with which NAF, or type of NAF (e.g. presence, certificate server), it shares a particular key (lack of key separation). But this could be important to know for UE if the key shared with the NAF is to be used for mutual authentication between UE and NAF, and the UE wants to make certain actions dependent on the (type of) NAF. Remember that no requirements have been put on the use of the shared key in protocol B. The bootstrapping architecture is just meant to provide shared keys to entities (UE and NAF) who initially do not share keys. The architecture should not restrict the uses of the shared key.

This contribution suggests an improvement of the currently specified procedure which consists in performing a key derivation, including NAF-specific parameters, before a key is sent from the BSF to a NAF.

1.2 Key update initiated by NAF

The current text of the TS is clear about how a UE can trigger the generation of a new key shared with a NAF: it simply invokes protocol A again, and then invokes protocol B with the new key identifier. The current text is not clear, however, how a NAF could trigger a key update. But it is certainly not acceptable that the UE unilaterally determines which key is used between the UE and the NAF. Therefore, a corresponding mechanism is needed.

2. Proposal

2.1 Key derivation

The following procedure is proposed:

A UE may communicate with a number of NAFs over time. The first time after power-up the UE wants to access any one of these NAFs the UE initiates protocol A with the BSF, and obtains a key Ks[1] shared with the BSF. The user then invokes protocol B with the NAF, sending a key identifier (assumed to be implied in the transaction identifier according to the TS[2]) along which allows the NAF and the BSF to uniquely identify the key to be shared between UE and NAF. The NAF then invokes protocol D to retrieve the key shared with the UE from the BSF, using the key identifier received from the UE. According to the current text of the TS, this key shared between the UE and the NAF is Ks.

The only difference between the current text of the TS and the proposal in this contribution is that the key shared between UE and a NAF is not Ks, but a NAF-specific key Ks_NAF derived from Ks using suitable input (e.g. RAND, User_Id, NAF_ID, cf. TD S3-030219). Ks_NAF may be specific to a particular NAF or a particular NAF-type (e.g. presence server or certificate server). One Ks may be used to derive only one key Ks_NAF or several such keys Ks_NAF for different NAFs or types of NAFs. If only one Ks_NAF is derived from one Ks then there is, of course, no reduction in the number of runs of protocol A. This may still be required in applications with very high security requirements (e.g. certificate requests). But if several Ks_NAF keys are derived from one Ks then such a reduction is achieved. This may be used in applications with lower security requirements (e.g. chat servers).

It could also be envisaged that, in future releases of the USIM, the key Ks remains in the UICC, and the key derivation is performed on the UICC. This is, however, not part of this contribution, and should not be mandated even in future releases.

The UE now proceeds by deriving this key Ks_NAF.

The procedure then continues as in the current text: the UE invokes protocol B with the NAF, sending a key identifier which includes RAND (possibly also User_Id and NAF_ID). The NAF invokes protocol D with the BSF to retrieve the key shared with the UE. Upon receiving the request from the NAF, the BSF derives the key Ks_NAF from Ks and sends it to the NAF.

2.2 Key update

The requirements put on protocol B need to be as small as possible so that the architecture is sufficiently generic and a large number of protocols can be used as protocol B. The current text of the TS makes it explicit that protocol B must be able to carry a transaction identifier which can be used to identify the key shared between UE and NAF. The NAF may want a key update from time to time. Therefore the NAF needs a means to indicate to the UE that a key update is desired. Protocol B may provide a means to explicitly indicate a desired key change. Whenever such a means is available it shall be used. But we should not mandate this. The minimum requirement on protocol B seems to be that the NAF can indicate a security violation to the UE. Receiving such an indication should then trigger the UE top invoke protocol A to obtain a new key shared with the BSF from which a new key shared with the NAF would then be derived.

3. Pseudo-CRs implementing section 2

The two proposals from the previous section are implemented in two different pseudo-CRs so as to allow separate discussion.

3.1 Pseudo-CR implementing section 2.1 (Key derivation)

[1] The shared key is assumed to be a function of CK and IK obtained during the run of protocol A. It is uniquely determined by the permanent key K and the challenge RAND.

[2] Section 4.3.2 of the TS: “ It is assumed that UE supplies sufficient information to NAF, e.g. a transaction identifier, to allow the NAF to retrieve specific key material from BSF.”

Text from TS ab.cde v020 (S3-030317)

4.3 Procedures

This chapter specifies in detail the format of the bootstrapping procedure that is further utilised by various applications. It contains the AKA authentication procedure with BSF, and latter the key material generation procedure.

4.3.1 Bootstrapping procedures

~~When a UE wants to interact with an NAF~~, . The first time after power-up the UE wants to access any one of the NAFs he may communicate with it shall first perform a bootstrapping authentication (see Figure 3):Editor's notes: Protocol C related procedure will be added here in future development. It may re-use Cx interface that is specified in TS 29.228. If the UE already shares a key with the BSF there is no need for the UE to perform a bootstrapping authentication unless the UE desired an update of that key or the UE has received a key update indication from the NAF.

[figure to be updated]

Figure 3: The bootsrapping procedure

1: The UE sends an HTTP request towards the BSF.

2. BSF retrieves the user profile and a challenge, i.e. the Authentication Vector (AV, AV = RAND||AUTN||XRES||CK||IK) by protocol C from the HSS.

3. Then BSF forwards the RAND and AUTN to the UE in the 401 message (without the CK, IK and XRES). This is to demand the UE to authenticate itself.

4: The UE calculates the message authentication code (MAC) so as to verify the challenge from authenticated network; the UE also calculates CK, IK and RES. This will result in session keys IK and CK in both BSF and UE.

5. The UE sends request again, with the Digest AKA RES as the response to the BSF.

6: If the RES equals to the XRES that is in the AV, the UE is authenticated.

7. The BSF shall send 200 OK message to the UE to indicate the success of the authentication.

8. The key material Ks is generated in both BSF and UE by concatenating CK and IK. The Ks is used for deriving further NAF-specific (or NAF-type specific) keys Ks_NAF. These NAF-specific keys are used to secure ~~securing~~ the protocol B.

Editor's note: ~~The key material~~ Ks as well Ks_NAF is 256 bits long. It is up each NAF to ~~make~~ adapt ~~the usage of~~ the key material Ks_NAF specifically according to the requirements of protocol B.

9. BSF may supply a transaction identifier to UE in the cause of protocol A.

4.3.2 Procedures using bootstrapped Security Association

After UE is authenticated with the BSF, every time the UE wants to interact with an NAF the following steps are executed as depicted in Figure4:

UE starts protocol B with the NAF

- In general, UE and NAF will not yet share the key(s) required to protect protocol B. If they already do, there is no need for NAF to invoke protocol D.

- It is assumed that UE supplies sufficient information to NAF, e.g. a transaction identifier, to allow the NAF to retrieve specific key material from BSF.
- The UE derives Ks_NAF from Ks and adapts Ks_NAF in order to obtain the keys required to protect protocol B ~~from the key material~~.

NAF starts protocol D with BSF

- The NAF requests key material corresponding to the information supplied by the UE to the NAF (e.g. a transaction identifier) in the start of protocol B.
- The BSF derives Ks_NAF from Ks and supplies to NAF the requested key material.
- The NAF ~~derives~~ adapts Ks_NAF in order to obtain the keys required to protect protocol B ~~from the key material~~ in the same way as the UE did.

NAF continues protocol B with UE

Once the run of protocol B is completed the purpose of bootstrapping is fulfilled as it enabled UE and NAF to run protocol B in a secure way.

[figure to be updated]

Figure 3: The bootstrapping usage procedure

Editor's note: Message sequence diagram presentation and its details will be finalized later.

3.2 Pseudo-CR implementing section 2.2 (key update)

Text from TS ab.cde v020

4.3 Procedures

This chapter specifies in detail the format of the bootstrapping procedure that is further utilized by various applications. It contains the AKA authentication procedure with BSF, and latter the key material generation procedure.

4.3.1 Bootstrapping procedures

When a UE wants to interact with an NAF, it shall first perform a bootstrapping authentication (see Figure 3):Editor's notes: Protocol C related procedure will be added here in future development. It may re-use Cx interface that is specified in TS 29.228. It shall also perform a bootstrapping authentication when it has received a key update indication from the NAF (cf. section 4.3.2).

[figure to be updated]

Figure 3: The bootsrapping procedure

1: The UE sends an HTTP request towards the BSF.

2. BSF retrieves the user profile and a challenge, i.e. the Authentication Vector (AV, AV = RAND||AUTN||XRES||CK||IK) by protocol C from the HSS.

3. Then BSF forwards the RAND and AUTN to the UE in the 401 message (without the CK, IK and XRES). This is to demand the UE to authenticate itself.

4: The UE calculates the message authentication code (MAC) so as to verify the challenge from authenticated network; the UE also calculates CK, IK and RES. This will result in session keys IK and CK in both BSF and UE.

5. The UE sends request again, with the Digest AKA RES as the response to the BSF.

6: If the RES equals to the XRES that is in the AV, the UE is authenticated.

7. The BSF shall send 200 OK message to the UE to indicate the success of the authentication.

8. The key material Ks is generated in both BSF and UE by concatenating CK and IK. The Ks is used for securing the protocol B.

Editor's note: The key material Ks is 256 bits long. It is up each NAF to make the usage of the key material specifically.

9. BSF may supply a transaction identifier to UE in the cause of protocol A.

4.3.2 Procedures using bootstrapped Security Association

After UE is authenticated with the BSF, every time the UE wants to interact with an NAF the following steps are executed as depicted in Figure4:

UE starts protocol B with the NAF

- In general, UE and NAF will not yet share the key(s) required to protect protocol B. If they already do, there is no need for NAF to invoke protocol D.
- If the NAF shares a key with the UE, but an update of that key it sends a suitable key update request to the UE and terminates protocol B. The form of this indication may depend on the particular protocol B and is ffs.
- It is assumed that UE supplies sufficient information to NAF, e.g. a transaction identifier, to allow the NAF to retrieve specific key material from BSF.
- The UE derives the keys required to protect protocol B from the key material.

NAF starts protocol D with BSF

- The NAF requests key material corresponding to the information supplied by the UE to the NAF (e.g. a transaction identifier) in the start of protocol B.
- The BSF supplies to NAF the requested key material. If the key identified by the transaction identifier supplied by the NAF is not available at the BSF, the BSF shall indicate this in the reply to the NAF. The NAF then indicates a key update request to the UE.
- The NAF derives the keys required to protect protocol B from the key material in the same way as the UE did.

NAF continues protocol B with UE

Once the run of protocol B is completed the purpose of bootstrapping is fulfilled as it enabled UE and NAF to run protocol B in a secure way.

Editor's note: Message sequence diagram presentation and its details will be finalized later.

[figure to be updated]

Figure 4: The bootstrapping usage procedure