3GPP TSG SA WG3 Security#30 **S3-030552**
6 – 10 October, 2003
Povoa do Varzim, Portugal

Agenda Item:	**7.9 (GAA)**
Source:	**Siemens**
Title:	**Key separation in a Generic Bootstrapping Architecture**
Document for:	**Discussion and decision**

Abstract

This contribution proposes to enhance the current procedure to distribute shared keys to a UE and a NAF defined in TS 33.109 by allowing the derivation of shared keys, which can only be used with a specific NAF or a group of NAFs.

1. Problem statement

Establishment of key shared between UE and NAF: The draft TS 33.109 “Bootstrapping of application security using AKA and support for Subscriber Certificates” (latest version in S3-030488) 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 (http digest aka) between the UE and a Bootstrapping Server Function (BSF). The procedures using bootstrapped security associations are described in section 4.3.2 of TS 33.109. 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. In the final message of protocol A, the BSF sends the UE a transaction identifier. The UE sends the transaction identifier to the NAF in protocol B. The NAF then retrieves the key sending the transaction identifier to the BSF in protocol D. This transaction identifier carries no information about the NAF. The BSF will give the key to any NAF which presents the transaction id and which authenticates to the BSF as a legitimate NAF. The BSF has no clue for which NAF the key is intended by the UE.

This has the following consequence:

Lack of key separation: the UE does not know with which NAF, or type of NAF (e.g. presence, PKI portal), it shares a particular key. But this could be important for the UE to know if the key shared with the NAF is used for authentication of the NAF to the UE, and the UE wants to perform certain actions only with a certain (type of) NAF. The bootstrapping architecture is just meant to provide shared keys to entities (UE and NAF) who initially do not share keys in a generic way. The architecture should not restrict the uses of the shared key. So it should be allowed that the shared key is used for NAF to UE authentication. A Generic Authentication Architecture should not be limited to server authentication by certificates.

An example where the bootstrapped key is used for mutual authentication between UE and NAF would be its use with shared_key_TLS, which was recently discussed on the SA3 mailing list.

The rest of the contribution is organised as follows: section 2 goes into some more detail regarding the threats when there is no key separation. Section 3 suggests an improvement of the currently specified procedure. The improvement consists in performing a key derivation, including NAF-specific parameters, before a key is sent from the BSF to a NAF. Section 4 discusses possible key derivation parameters. Section 5 proposes a flexible mechanism to signal information about the key derivation scheme.

2. Threats when key separation is lacking

A NAF may be just any type of application server. It may be e.g. a presence server, or a PKI portal, or a conference server, or a game server. It may reside in the operator’s home network, or in a visited network. The applications provided by the NAF may be defined outside 3GPP, e.g. by OMA or other standards organisations, or may even be proprietary. Application servers may be provided by a large variety of different vendors. Some applications may have to be introduced quickly by operators to satisfy market needs. As a consequence, it can be expected that the security levels of NAF implementations, and the security of the operating environment of NAFs, will vary considerably.

The main threats to the current bootstrapping architecture in TS 33.109 arise when the shared key is used for NAF to UE authentication., e.g. in shared_key_TLS. Due to the possibly different security levels of NAFs, it may happen that one of the many NAFs in one network (not necessarily the home network) is successfully attacked, either by insiders, e.g. administrators, or remotely by hackers. An attacker controlling a NAF could then retrieve just any key from the BSF whose transaction identifier is known to him. A rogue NAF could in this way impersonate any other NAF known to a particular BSF as legitimate, without a possibility for the user to know.

In this way, a security breach in one type of application (e.g. through a software glitch) may then spread to other types of applications, and a security breach in one network may be exploited to perform attacks in other networks.

But, in general, it is important for the user to know to which NAF he is talking. E.g., a user may want to entrust certain personal data to a PKI portal of his home network, but perhaps not to a commerce server in a visited network.

It would therefore be prudent to design a Generic Bootstrapping Architecture in such a way that a spreading of security failures across the entire system can be prevented by subdividing the system in compartments which are mutually separated from each other from a key distribution point of view.

Examples of attacks (other types of attacks may be possible):

Variant1: a user may want to talk to an application server with the name PKI_portal.com. Now assume that one application server NAF1 was successfully attacked, and that the attacker has control over interface D from NAF1 to the BSF. The attacker now sets up a rogue server with the name PKI-portal.com, and may be able to entice a user to connect to it. As social engineering goes, the user will not notice, or not care about, the minimal difference in the names of the two PKI portals. Note that the attacker needs no control over the routing path between the UE and the rogue server, as the attacker could legitimately obtain an entry in the DNS for the modified server name, and the DNS could return an entirely different IP address for the modified server name than for the original name. The attacker can eavesdrop on the messages of protocol A between UE and BSF, or the attacker can simply wait for the first message from the UE in protocol B, and in this way the attacker can learn the transaction identifier. The attacker then triggers NAF1 to retrieve the shared key corresponding to the transaction identifier from the BSF, and provides the rogue "PKI-portal.com" server with that key. The UE and the rogue server will then successfully perform mutual authentication.

Variant2: here, it is assumed that the attacker can play man-in-the-middle on the first link from the UE, and, in this way, has control over the routing path from the UE to the server. This would be the case e.g. when the attacker used a false base station to which the UE attached over the radio link. Then the attacker could even use the true server name PKI_portal.com towards the user. The rest of the attack works as for variant 1.

The attack can be prevented for both variants if

- interface D is assumed to provide mutual authentication, confidentiality and integrity; and
- the BSF checks the NAF identity against a list of authorised NAFs; and
- the key shared between UE and NAF is specific to one NAF or a defined set of NAFs.

The attack is prevented because, in both variants, the attacked server NAF1 would be able retrieve only keys from the BSF, which are specific to NAF1or the set to which NAF1 belongs. The keys could be used with neither PKI-portal.com nor PKI_portal.com.

3. Key derivation

In order to mitigate the threats identified in section 2, the following key derivation procedure is proposed:

In the current text of TS 33.109, the key shared between a UE and a NAF is Ks, the key resulting from a run of protocol A between UE and BSF. It is proposed in this contribution that a NAF-specific key Ks_NAF is used between UE and a NAF instead. This NAF-specific key is derived from Ks using suitable input parameters.

Ks_NAF may be specific to a particular NAF or a set of NAFs. This is ffs and is determined by the type of key derivation parameters. Alternatives are discussed in section 4. The derived key is computed as

Ks_NAF = H(Ks, Key_deriv_par) where Key_deriv_par is the key derivation parameter which characterises the NAF, or set of NAFs, and H is a suitable key derivation function.

The procedure would then be as follows:

- the UE runs protocol A with the BSF, as in the current text of TS 33.109. During the protocol run, key Ks is established, and the UE receives a transaction identifier;
- the UE invokes protocol B with the NAF, sending the transaction identifier received in protocol A.
- The NAF invokes protocol D with the BSF to retrieve the key shared with the UE, sending the transaction identifier received in protocol B. Upon receiving the request from the NAF, the BSF derives the key Ks_NAF from Ks and the key derivation parameter, and sends it to the NAF.
- The UE derives the key Ks_NAF from Ks and the key derivation parameter, and uses it in protocol B.

(Only the marked text deviates from the current TS 33.109.)

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.

4. Key derivation parameters

The key derivation parameters Key_deriv_par are expected to include:

- the user identity or the transaction identifier (The user identity may be part of the transaction identifier.);
- a NAF identifier (see below);
- possibly additional random information (e.g. the RAND from the run of protocol A).

For a discussion of suitable key derivation parameters in a slightly different context see also the LS from SAGE in S3-030219.

The rest of this section deals with the question which type of NAF identifier should be included. This type of NAF identifier determines the degree of assurance the user gets about the identity of the NAF-server it talks to through the mutual authentication using Ks_NAF.

We consider several alternatives for the NAF identifier to be included in the key derivation parameter:

1. *DNS server name*: this would be the finest granularity. The UE would use the DNS name used in requests in protocol B (e.g http requests). The BSF would use the FQDN from the DIAMETER request in protocol D.
   In the case of shared_key_TLS, this solution would imply that, in the case of name-based virtual hosts on the same machine (e.g. a reverse proxy), several different TLS connections would have to be used between the UE and the NAF (e.g. a reverse proxy), one for each server name, because the session keys would be different for each virtual host. But this apparent shortcoming may not be specific to the use of the shared key for NAF-to-UE authentication, because a companion contribution by Siemens to this meeting shows that the same problem exists, when server certificates are used for NAF-to-UE authentication. So this drawback may be difficult to avoid even in a more general setting.

But it is true that there may be situations where the user does not care about which server in particular he is talking to, as long as he knows that the server is authorised, e.g. the user's home network provider, to provide a particular service. So, a coarser granularity of server authentication may suffice. This leads to the points 2 and 3 below.

2. *Higher-level domains in the DNS server name*: only the n (e.g. two or three) highest domains in the server name would be used as NAF identifier, not the full DNS name. E.g in a NAF server name "server1.presence.bootstrap.operator.com", the server-side key derivation parameter would be "operator.com " or "bootstrap.operator.com ". The UE and the BSF would have to be explicitly told which scheme to use, and it would probably be difficult to make updates or changes, unless there is flexibility built into the system from the start. A flexible solution is outlined in section 5.
   With this type of NAF identifier, in the case of shared_key_TLS, several name-based virtual hosts would be able to use the same key, so, in principle, only one TLS connection between a UE and a reverse proxy could suffice. But

this advantage may be theoretical only, as it is unclear how a UE would know which servers would be served by the same reverse proxy.

3. *Types of NAFs or application names*: DNS server names of NAFs would be mapped to pre-defined types or application names, which would be used as NAF identifiers. The UE and the BSF would have to be told which DNS server names are grouped with which type or application name. 3GPP would have to standardise a format for, and a complete list of, types or application names. The configuration effort for updates in the UE would be probably be considerable, if not prohibitive.
   Regarding shared_key_TLS, the same remark as for item 2 above applies.

4. *NAF IP address*: several DNS server names could map to the same IP address of a NAF, and the IP address would be used as NAF identifier.
   Regarding shared_key_TLS, there would be the possibility to have only one TLS connection between UE and a NAF, which is a reverse proxy. But the fundamental problem with this approach is that the UE may sit behind a forward proxy and may never know the IP address of the server. Therefore, this approach is considered infeasible.

From the four alternatives above, only 1 and 2 are proposed for further study. Alternative 3 is considered to come with too much configuration effort, and alternative 4 has the problem with forward proxies.

In the next section we show how sufficient flexibility could be built into the system so that both, alternatives 1 and 2, could be accommodated.

5. Flexible signalling of key derivation schemes

The last message in protocol A is a 200 OK from BSF to UE, which includes the transaction identifier. In TD S3-030341, Nokia proposed an encoding of the transaction identifier as an XML document in the HTTP response payload. The transaction identifier could be extended to include a “key derivation scheme identifier” KEY_DER_ID. Remember that KEY_DER_ID would be integrity-protected by http digest as it is part of the http payload.

It is believed that at most one byte would suffice for KEY_DER_ID. We show the effect of a particular value of KEY_DER_ID on the NAF identifier which is input to the key derivation parameter, by using the example of a UE accessing a NAF with server name “server1.presence.bootstrap.operator.com".

KEY_DER_ID could take the following values:

KEY_DER_ID = 0: this means no key derivation. The key shared between UE and NAF is Ks.

KEY_DER_ID = 1: this means that only the highest domain in the DNS server name is input to the key derivation. The NAF identifier is “com”.

KEY_DER_ID = 2: this means that the two highest domains in the DNS server name is input to the key derivation. The NAF identifier is “operator.com”.

KEY_DER_ID = 3: this means that the three highest domains in the DNS server name is input to the key derivation. The NAF identifier is “bootstrap.operator.com”.

Etc.

KEY_DER_ID = 7: this means that ALWAYS the FULL DNS server name is input to the key derivation (even if there should be more than seven domains in the server name). The NAF identifier is “server1.presence.bootstrap.operator.com".

In this way, flexibility could be built into the system so that several alternatives for key derivation could be accommodated. But this would, of course, somewhat increase the complexity of the system. It is ffs whether the full flexibility is needed.

Conclusions and Proposals

1. The Generic Bootstrapping Architecture is meant to be a generic tool to provide shared keys to UEs and application servers, independent of particular key uses. Section 2 showed threats, which become possible if only one application server is successfully attacked. SA3 is asked to endorse that the threats should be mitigated by appropriate provisions in the standard. In particular, it shall be prevented that a security breach in one application server can spread across the entire system.

2. It was proposed in section 3 to limit the effect of a security breach in one part of the system to a small part of the system by introducing key derivation for the keys shared between UE and NAF. SA3 is asked to endorse the use of a suitable key derivation procedure.

3. Section 4 proposed certain alternatives for the NAF identifier which is input to the key derivation parameters. The NAF identifier would determine the degree of assurance the UE gets about the identity of the NAF in NAF-to-UE authentication. SA3 is asked to agree to study only alternatives 1 (use DNS server name) or 2 (use defined parts of DNS server name) further and select between these alternatives at the next meeting.

4. Section 5 proposed a flexible mechanism to signal that one out of possibly multiple key derivation schemes be used with a certain key. As a minimum, the mechanism could be used to signal whether no key derivation or some pre-determined key derivation is used. SA3 is asked to endorse the use of this flexible signalling mechanism.