3GPP TSG SA WG3 Security – S3#23 **S3-020189**

14 - 17 May 2002

Victoria, BC, Canada

Source: **Gemplus Card International, Oberthur Card Systems**

Title: **Role of UICC in secure PKI architectures**

Document for: **Discussion**

Agenda Item: **T.B.D**

Abstract

This input paper aims at proposing a pertinent way of defining a PKI architecture and exploiting it in a secure manner with the help of the UICC.

1. Technical Proposal

In order to ensure a reasonable level of security for end-to-end transaction, public key based signature schemes have to be used; to ensure the validity of the differents keys, we need the support of a PKI infrastructure.

Requirements:

- To ensure a high level of security, the secret (symmetric and asymmetric) keys have to be stored in the UICC; this implies that signature has to be performed in the UICC.
- To sign a document requires a high level of confidence in the authenticity of the other actors, which is achieved by certification using the PKI. In order to keep this high level of security throughout the chain, we need to check the complete chain of certification in the signing entity, which is the UICC. Furthermore, the link between the actors and the certified and checked public keys has to be kept in the UICC.
- There has to be no size constraints on the depth of the PKI hierarchy.

We propose to achieve this by implementing the following functions:

Reset_current_key ():
initializes the current_key object to the root Certification Authority public key.

Update_current_key (certificate):
checks, with the current_key, the certificate given as input. If OK, replaces current_key by the public key contained in the certificate.

Store_validated_certificate ():
stores, in the certificate database of the UICC, the current_key (and additional info to be defined) associated with a validity date which is the nearest date in the PKI chain, and which might be shorten by specific operator policy.

The UICC certificate database is of fixed size. The storage and access policy has to be defined (keeps last n certificates stored, or fills up until full and provides a means to delete certificates).

Set_current_key (cert_number):
checks a certificate stored in the card (identified by the cert_number) in regard with the validity date associated by the card (see store_validated_certificate() function). If OK, the current_key is the stored public key of this certificate.

Before the execution of following commands dealing with signatures or certificates, the Setup_key function is performed to indicate the public key involved.
Setup_key (pubkey):
- If pubkey field is empty then the public key to use for following commands will be the current_key.
- Else the public key will be the pubkey value provided as input.

The case “pubkey field empty” is more secure because the chain of certification was checked.

Verify_certificate (certificate, time):
checks the validity of the certificate in regard with the signature (using the key defined in setup_key) and the time.

Sign_text (text):
returns the signature of the text given as input.

Verify_signature (signed_text):
verifies the signed_text (using the key defined in setup_key)

Sign_verified_text (strip, signed_text):
first verifies the signed_text using the current_key, then
if the verification is OK,
- if strip is no, formats and signs the whole signed_text message (including the signature) and return the newly signed message.
- if strip is yes, formats and signs the signed_text message after having removed the signature information and return the newly signed message

If the verification is not OK, returns an error message.

Get_card_certificate ():
returns the certificate of the public key of the card signed by the CA.

Additionally, the UICC can provide functionalities to manage certificate life cycle. These functionalities are the following:

GenerateKeyPair():
generates a key pair (public key, private key) for a public key scheme on board the card, keeping the private key securely stored in the card.

SetUpCertificate():
stores securely a certificate in the UICC memory.

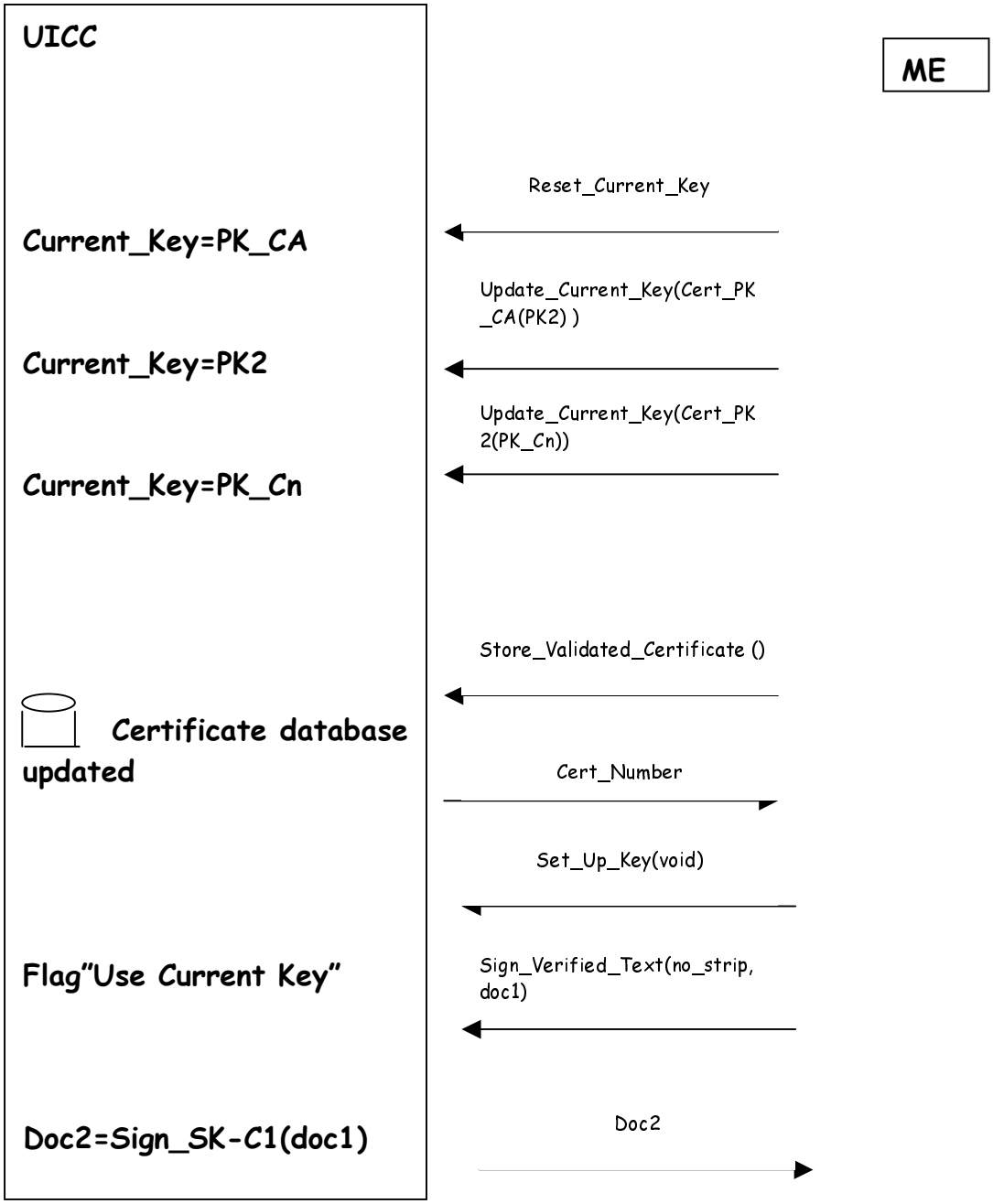
RevokeCertificate():
flags a certificate as no longer usable.

UpGradeCertificate():

replaces an existing certificate by a new version (equivalent to a DeleteCertificate followed up by a SetUpCertificate).

DeleteCertificate():
removes the certificate from memory

An example of process is:



2. Advantages

2.1. For the end-user

Absolute security, as everything is computed in the card.

2.2. For the operator

Large possibilities of cross-certification and interoperability, since there is no limitation on the size of the PKI.

2.3. For the service provider

Easy to get certified by any operator.

3. Arguments

We focus on the advantages of proposing security services in the UICC.

- **Tamper-resistance**: UICC is clearly identified as the privileged tamper-resistant part of the mobile client. Therefore, we find it mindful to generate asymmetric key pairs on board the card, to keep their private part on board, and to export a properly signed copy of their public part to an external entity ;

- **Coherency with other works**: such functions as SignText are in line with the Wap Forum working assumptions ;

- **EMV compatibility for m-commerce:**

- **Maturity of on board asymmetric key pair generation (OBKG)**: this technology is mature and can be deployed on a large scale ;

- **UICC as an active cryptographic device**: public key signature production and verification algorithms have been available on board smart cards for quite a long time;

- **High storage reliability**: due to smart card high quality memory management, there exists active integrity checks in the card memory. It turns out that smart card operations are naturally atomic.

- **Strong USIM-Operator link**: the AKA is an obvious operator-USIM link. There, the operator is responsible for the management of the PKI, especially the CA part; to ensure end-to-end security in the system, the keys have to be generated and authenticated in the security domain of the operator. The obvious candidate on the client side is then the USIM.