CR-Form-v7

CHANGE REQUEST

⌘ **S3-040272** **CR** **CRNum** ⌘ **rev** **-** ⌘ Current version: **6.0.0** ⌘

*For **HELP** on using this form, see bottom of this page or look at the pop-up text over the ⌘ symbols.*

Proposed change affects: UICC apps⌘ [X] ME [ ] Radio Access Network [ ] Core Network [ ]

Title:	⌘	Session Key Exchange Algorithm (SKEA) for Local Interface Trusted Tunnel Establishment
Source:	⌘	Intel
Work item code:	⌘	WLAN-WWAN Interworking
Date:	⌘	28/06/2004
Category:	⌘	**B**
Release:	⌘	Rel-7

Use one of the following categories:
F *(correction)*
A *(corresponds to a correction in an earlier release)*
B *(addition of feature),*
C *(functional modification of feature)*
D *(editorial modification)*
Detailed explanations of the above categories can be found in 3GPP TR 21.900.

Use one of the following releases:
2 (GSM Phase 2)
R96 (Release 1996)
R97 (Release 1997)
R98 (Release 1998)
R99 (Release 1999)
Rel-4 (Release 4)
Rel-5 (Release 5)
Rel-6 (Release 6)

Reason for change:	⌘	S3-040272 briefly mentions possible use of Shared Keys for authentication in the Local Link Trusted Tunnel. In Section 3.6. This CR adds more detailed material in a new Annex A.
Summary of change:	⌘	In the context of Local Interface Trusted Tunnel, SKEA works with an adaptation of the TLS protocol to provide per packet protections. SKEA is provided as part of a new TLS Ciphersuite for Shared Secret based authentication and TLS Master Secret generation. Thus it maintains compatibility with the TLS protocol..
Consequences if not approved:	⌘	The use of certificates for Local Link Trusted Tunnel can cause performance & provisioning issues. SKEA can enhance the performance & save compute resources at the UICC/Reader without affecting security, and therefore can make the Trusted Tunnel a stronger & more convenient solution.

Clauses affected:	⌘	3.6
Other specs affected:	⌘	Y N [ ] [ ] Other core specifications [ ] [ ] Test specifications [ ] [ ] O&M Specifications
	⌘	
Other comments:	⌘	None.

***** Start of change *****

Annex A

Session Key Exchange Algorithm (SKEA) for Local Interface Trusted Tunnel Establishment

Introduction

SKEA is designed to work with an adaptation of the TLS protocol [4] to provide per packet protections. SKEA is intended to be provided as part of a new TLS Ciphersuite for shared secret based authentication and TLS Master Secret generation. Thus, it maintains full compatibility with the TLS protocol as defined in reference [4].

The TLS Protocol as defined in [4] requires the use of Public Key Certificates at the client and the server to accomplish authentication (mutual authentication is optional) and Master Secret generation. However it does provide flexibility to use other methods for authentication & per packet protections. The SKEA does not require Public Key Certificates for authentication or Master Secret generation. Proposals generated within the IETF for using the TLS "Session Resumption" mechanism allow a session to be established without the expensive part of the certificate based handshake based on public key cryptography. Reference [2] describes the use of the TLS session resumption capability to set up a protected channel without the extra overhead required in session initialization based on public key methods.

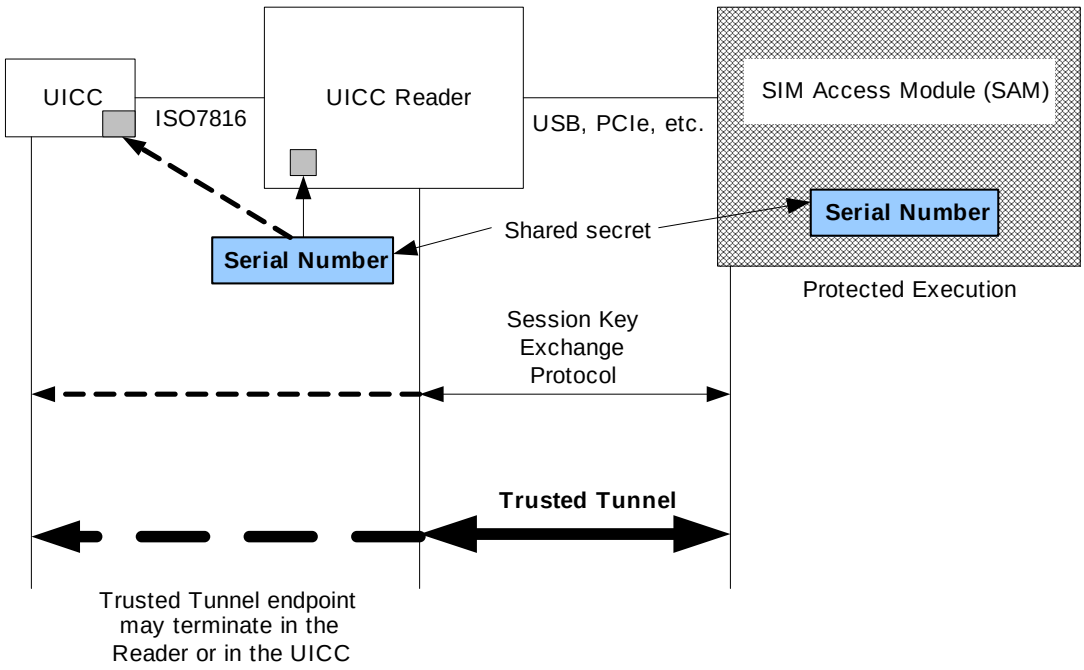

Figure 1: Trusted Tunnel Overview

The main goal of SKEA is to set up a Session Key (*K*) at both the Software Client on the terminal (SIM Access Module – SAM) and the UICC/Reader. SKEA must run both at the terminal's software client and at the UICC/Reader. It is assumed that the SKEA runs in a suitably protected execution environment, the specifics of which are outside of the scope of this contribution.

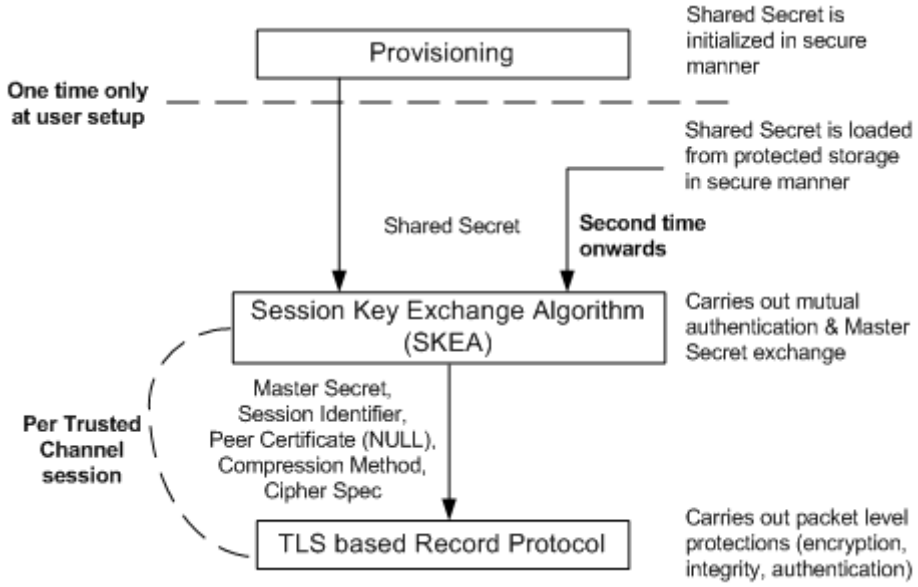

Figure 2: Protocol Overview

Algorithm Advantages

The main advantages of SKEA are:

- Provides mutual authentication between Terminal and UICC/Reader [5]
- Use of symmetric key authentication eliminates the need for RSA or other public key methods, which tend to be slower. This saves crypto hardware at the UICC/Reader
- Use of symmetric key methods eliminates the need to address provisioning for public keys / certificates.
- Capability to produce different Trusted Tunnel TLS Master Secret (***K***) for each session.
- Leverages accepted mechanisms for key generation using a shared secret.
- Hashing algorithm in final step can be adjusted to get a session key of desired size
- AES is believed to provide better security over other encryption methods.

Algorithm Description

This algorithm is based on Reference [2] with enhancements described in Reference [1]. Figure 1 below depicts how this exchange works with the SON architecture. For the figure and the description below, SHA-X is assumed to be SHA-1 for the sake of simplicity.

A. The UICC/Reader and the Software Client share a secret, ***S*** (160-bit, 32 characters base32 encoded).
  a. The secret ***S*** is pre-shared via sticker or on the case other hardcopy and the owner types it into the Software Client or similar. To prevent attacks by a malicious user, the secret ***S*** can also be installed by the service provider, e.g. through a secure website

B. At the start of the protocol, the Software Client.
  a. Generates a random nonce, N_{SAM} (160-bit)

b. Sends $\boldsymbol{N_{SAM}}$ to the UICC/Reader.

C. The UICC/Reader then
- **a.** Generates random nonce $\boldsymbol{N_{READER}}$ (160-bit)
- **b.** Computes $\boldsymbol{AUTH_{READER}}$ = SHA-X($\boldsymbol{S}$ | $\boldsymbol{N_{READER}}$ | $\boldsymbol{N_{SAM}}$)
- **c.** Sends both $\boldsymbol{N_{READER}}$ and $\boldsymbol{AUTH_{READER}}$ to the SAM.

D. The Software Client
- **a.** Checks $\boldsymbol{AUTH_{READER}}$ that was sent by the UICC/Reader. This authenticates the reader.
- **b.** Computes $\boldsymbol{AUTH_{SAM}}$ = SHA-X($\boldsymbol{S}$ | $\boldsymbol{N_{SAM}}$ | $\boldsymbol{N_{READER}}$)
- **c.** Sends $\boldsymbol{AUTH_{SAM}}$ to the reader.

E. The UICC/Reader
- **a.** Checks $\boldsymbol{AUTH_{SAM}}$. This authenticates the Software Client. Mutual authentication is complete

F. To compute the session key $\boldsymbol{K}$, both parties compute
$\boldsymbol{x}$ = SHA-X($\boldsymbol{N_{READER}}$ | $\boldsymbol{N_{SAM}}$ | $\boldsymbol{S}$)
and use the most significant 128 bits of $\boldsymbol{x}$ as an AES key.

G. Both the Software Client and the UICC/Reader then initialize AES in counter mode, using the least significant 32 bits of $\boldsymbol{x}$ as the initial counter value (after padding to make total length 128 bits), and 48 bytes are generated for use as the **TLS master secret** $\boldsymbol{K}$.

H. Standard TLS client/server session key derivation is used from this point.

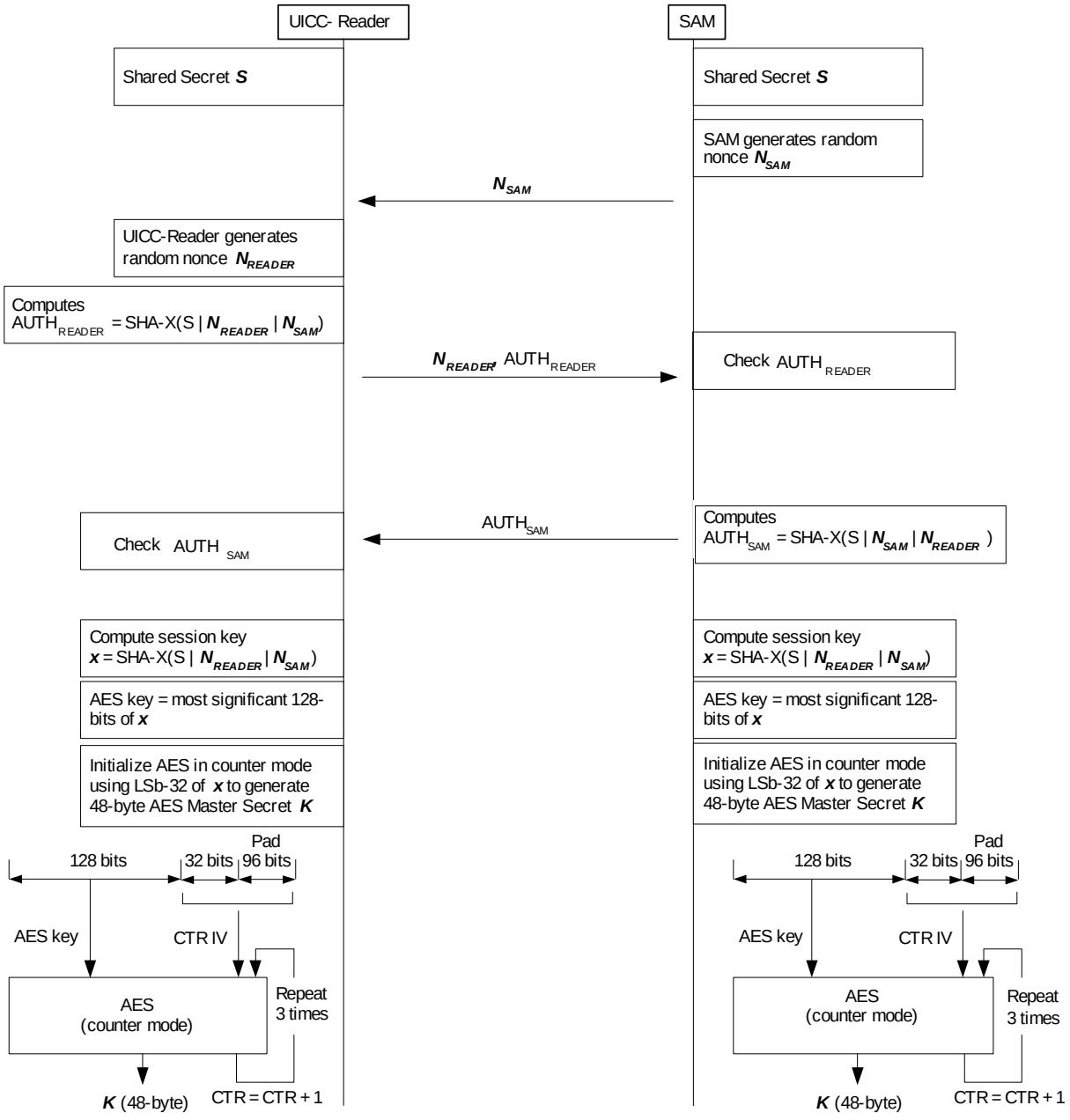

Figure 3: SKEA Algorithm

References

[1] "Using Advanced Encryption Standard (AES) Counter Mode With IPSEC Encapsulating Security Payload (ESP)", RFC 3686, January 2004, IETF, http://www.ietf.org

[2] "Use of Shared Keys in the TLS Protocol", Gutman, P., (October 2003), IETF, http://www.ietf.org/internet-drafts/draft-ietf-tls-sharedkeys-02.txt

[4] "Transport Layer Security" RFC 2246, IETF

[5] 3GPP S3-040272, Use of a Trusted Tunnel to Secure Local Terminal Interfaces.

***** End of change *****

**** END SET OF CHANGES *****