Cellular Authentication & Key Agreement for Service Providers

John A. MacDonald
Information Security Group
Royal Holloway, University of London,
Egham, England TW20 0EX
john@madgo.com

***Abstract*—This paper proposes an alternative to the 3GPP Generic Bootstrapping Architecture protocol for bootstrapping security credentials in mobile networks. The proposed protocol avoids certain privacy issues arising from the use of the 3GPP protocol, which may be of particular concern in e-health applications.**

I. Introduction

Marti et al. describe [13] e-health services that incorporate the use of mobile technologies to communicate with a Body Area Network (BAN) comprising diagnostic sensors and treatment dispensing actuators. There are serious user privacy concerns with the 3GPP Generic Bootstrapping Architecture (GBA) protocol when used as the basis for security for certain applications such as health.

This paper proposes an Authenticated and Key Agreement (AKA) protocol that reduces the privacy risk inherent in the 3GPP Generic Bootstrapping Architecture (GBA). Reduction of risk is achieved by both eliminating the mobile operator as a trusted third party and removing the dependency on the native GBA Bootstrapping Client in the mobile equipment (ME). Section 2 reviews the assumptions and requirements, whilst section 3 describes the proposed protocol. Section 4 lists the properties of the novel protocol, and compares it with the GBA architecture.

II. Prerequisites for Protocol

The proposed AKA protocol builds on the dual capabilities of SMS Security [1] and USIM Application Toolkit (SAT) [8]. The former provides end to end security services for an SMS message sent to or from the USIM card, whilst the SAT API allows a USIM card application to communicate with the host ME.

It is assumed that the ME provides a J2ME java runtime environment [16], complemented by additional classes from the Mobile Information Device Profile 2.0 specification [11] including the SATSA-APDU and SATSA-PKI [12] packages. These enable J2ME applications installed in the ME to access the tamper resistant USIM card using APDU communication, and provide support for digital signatures and management of security credentials. Java applications that run on MIDP compliant MEs are known as MIDlets. The USIM [3] is assumed to be a multi application UICC Java Card, and the USIM application [4] is just one possible java application [5].

It is further assumed that the USIM provides a random number generator function, and that the Mobile Operator which issued the USIM offers a delivery and installation service of MIDlets and Applets to UEs. To allow installation of MIDlets to the MIDP2.0 Operator domain of the ME, the Mobile Operator generates an asymmetric key pair and obtains a certificate Cert$_{MO}$ for the public key from a Certification Authority. The private key is used to digitally sign the MIDlet. Our protocol is based on the assumption that the ME has access to a trusted copy of the public key of the Certification Authority used to sign the Mobile Operator's public key certificate Cert$_{MO}$. We also assume that the Mobile Operator's certificate is in a format processable by the ME. The associated trust issues are beyond the scope of this paper.

III. Protocol

The end points of the proposed AKA protocol reside in java application space of the ME and the USIM. The protocol uses both symmetric and asymmetric cryptographic techniques [14] to provide the authentication and integrity services required, and is summarised in figure 1.

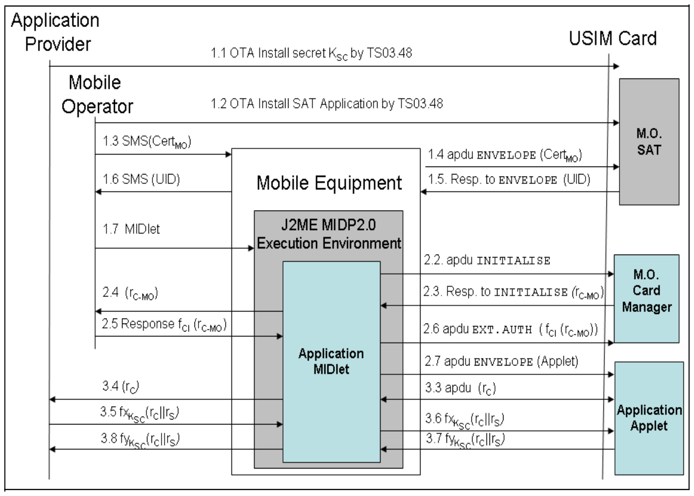
Application Provider
USIM Card
1.1 OTA Install secret K_{SC} by TS03.48
Mobile Operator
1.2 OTA Install SAT Application by TS03.48
M.O. SAT
1.3 SMS(Cert$_{MO}$)
1.4 apdu ENVELOPE (Cert$_{MO}$)
1.6 SMS (UID)
Mobile Equipment
1.5. Resp. to ENVELOPE (UID)
1.7 MIDlet
J2ME MIDP2.0 Execution Environment
2.2. apdu INITIALISE
M.O. Card Manager
2.4 (r_{C-MO})
2.3. Resp. to INITIALISE (r_{C-MO})
2.5 Response f_{CI} (r_{C-MO})
2.6 apdu EXT.AUTH (f_{CI} (r_{C-MO}))
Application MIDlet
2.7 apdu ENVELOPE (Applet)
3.4 (r_C)
3.3 apdu (r_C)
Application Applet
3.5 $fx_{K_{SC}}(r_C||r_S)$
3.6 $fx_{K_{SC}}(r_C||r_S)$
3.8 $fy_{K_{SC}}(r_C||r_S)$
3.7 $fy_{K_{SC}}(r_C||r_S)$

Fig. 1. Scheme Description

In this figure, the numbered protocol exchanges of the four phases refer to the detailed description below. We use the

PERVASIVEHEALTH 2008, 30 Jan - 1 Feb. Tampere, Finland
Copyright © 2008 ICST 978-963-9799-15-8
DOI 10.4108/ICST.PERVASIVEHEALTH2008.2571

following notation:

$$\begin{aligned}
MO &= \text{Mobile Operator and Card Issuer}\\
S &= \text{Application Server of e-health provider}\\
M &= \text{MIDlet application resident on ME}\\
C &= \text{USIM card}\\
K_{CI} &= \text{Card Manager shared secret key}\\
K_{SC} &= \text{e-health application shared secret key}\\
\text{Cert}_{MO} &= \text{Public key certificate of Operator domain}\\
\epsilon_K(D) &= \text{encryption of data D using key } K\\
\epsilon_{SAT}(D) &= \text{encryption of data D using 03.48 standard}\\
S(D, s_x) &= \text{Signature on data D using private key } s_x\\
\text{MAC}_K(D) &= \text{MAC of data D using secret key } K\\
\text{APDU}() &= \text{APDU command from MIDlet to USIM}\\
\text{AID}_n &= \text{USIM Card Manager identifier for Applet } n\\
\text{SMS}() &= \text{SMS communication mechanism}\\
s_x &= \text{private key corresponding to Cert}_{MO}\\
r_C &= \text{Random nonce generated by USIM } C\\
r_S &= \text{Random nonce generated by Server } S\\
i_S \;\&\; i_C &= \text{identifier of Server \& USIM respectively}\\
CK \;\&\; IK &= \text{Cipher \& Integrity Keys respectively}
\end{aligned}$$

PHASE 1: Install MIDlet in MIDP device.

The security requirement for this protocol phase is to install a MIDlet application on the ME with the necessary permissions to allow subsequent installation on the USIM of Applet bytecode.

1) $S \rightarrow C:$ SMS($\epsilon_{SAT}(K_{SC})$)
A long term secret key K_{SC} known only by the e-health application provider is securely copied to the USIM by the e-health provider using the TS03.48 mechanisms. This is used to provide end to end security [9], using credentials known only to the e-health provider.
2) $MO \rightarrow C:$ SMS(ϵ_{SAT}(SAT Application))
The payload of this SMS message from the Mobile Operator contains the SAT code to be installed on the USIM. Once installed, the SAT application uses proactive commands to register to be informed, via the ISO/IEC 7816-4 `ENVELOPE` APDU command , when a specific event occurs.
3) $MO \rightarrow M:$ SMS(Cert$_{MO}$)
A Domain Protection Root Certificate, Cert$_{MO}$, is sent to the ME by the Mobile Operator as the payload of a series of SMS messages .
4) $M \rightarrow C:$ APDU(`ENVELOPE`: Cert$_{MO}$)
The ME transfers the payload to the SAT application in the data field of an `ENVELOPE` APDU command. The Domain Protection Root Certificate is securely stored in the USIM Card.
5) $C \rightarrow M:$ APDU(`ENVELOPE`: UID)
The SAT application retrieves the USIM's unique identifier UID, and returns it to the ME as the response to the `ENVELOPE` command.
6) $M \rightarrow MO:$ SMS(UID)
The unique identifier is then returned to the Mobile Operator in the *Response Packet*. This acts as a proof of delivery and enables the Card Issuer (in our scenario the Mobile Operator) to fetch the specific Card Issuer Key K_{CI} of the USIM from the centralised database. Although a USIM card can execute multiple applets from different providers, in accordance with the Global Platform [6] specification, the Card Manager application guarantees the overall coherency of the multi-applet USIM card by ensuring that new Applets are integrity checked and their source authenticated prior to installation. The Card Manager application is provided by the entity which issued the card [7], [15].
7) $MO \rightarrow M:$ S(MIDlet,s_x)$||$(MIDlet)
Using the USIM's unique identifier, the Mobile Operator packages the e-health MIDlet application, which contains the appropriate install commands and the bytecode for the e-health Applet. The Applet bytecode is encrypted and integrity protected with a MAC using keys derived from the Card Issuer key K_{CI}, and packaged within the MIDlet JAR file. The MIDlet is signed using the private key corresponding to the public key contained within Cert$_{MO}$. This signature and certificates to validate the application are inserted within the application descriptor of the MIDlet and the JAR file transferred to the ME. The JAR file groups compiled Java code and associated metadata. The ME J2ME implementation verifies the signature using the public key in the certificate Cert$_{MO}$, and installs the MIDlet in the Operator domain of the MIDP2.0 runtime environment.

This protocol phase uses standardised MExE [2] techniques to authenticate the source and validate the integrity of the MIDlet JAR file to ensure installation in the ME with the required security permissions.

PHASE 2: Install Applet in USIM card

The security requirement for this protocol phase is to install the Applet bytecode in the USIM without threat of tampering.

1) $M \rightarrow C:$ APDU(`SELECT`: AID$_{CM}$)
The MIDlet now executes its Applet installation routine using APDU communications provided by the SATSA-APDU package. The MIDlet starts by using the `SELECT` command to select the USIM Card Manager application via its unique application identifier AID$_{CM}$, to initiate communication with the USIM Card Manager application.
2) $M \rightarrow C:$ APDU(`INITIALIZE UPDATE`)
Once the Card Manager application has been selected, the MIDlet issues a `INITIALISE UPDATE` command to advise the Card Manager that it wishes to install a new Applet onto the USIM.

3) $C \rightarrow M:$ APDU(r_{C-MO})
Before the Card Manager will accept installation of an Applet onto the SIM Card, it must first authenticate the source of the Applet. It does this by responding to `INITIALISE UPDATE` with a random number r_{C-MO}.

4) $M \rightarrow MO:$ r_{C-MO}
The MIDlet is not in possession of the Card Manager secret K_{CI} required to prove a trusted source for a new applet, so the challenge r_{C-MO} is sent back to the Mobile Operator (i.e. the USIM Card Issuer) by initiating an http session.

5) $MO \rightarrow M:$ $\epsilon_{K_{CI}}(K_{CI} \| r_{C-MO})$
The Card Issuer encrypts K_{CI} and r_{C-MO} with K_{CI} and returns the byte string to the MIDlet.

6) $M \rightarrow C:$ APDU(`EXT. AUTHEN.`: $\epsilon_{K_{CI}}(K_{CI} \| r_C)$)
The MIDlet requests the Card Manager to authenticate the source of the Applet by issuing the `EXTERNAL AUTHENTICATE` command providing the encrypted response to the random challenge r_{C-MO}. The Card Manager, also in possession of K_{CI}, authenticates the source of the Applet.

7) $M \rightarrow C:$ APDU(MAC$_{K_{CI}}$(Applet)$\|\epsilon_{K_{CI}}$(APPLET))
The MIDlet now transfers the encrypted and integrity protected Applet bytecode to the USIM card via the `ENVELOPE` command. At no point does the MIDlet have any knowledge of the key K_{CI} as it acts as a delivery mechanism between USIM and Server for predefined parcels of bytes. The integrity and confidentiality of the Applet code and the Card Issuer secret K_{CI} is assured. The Applet includes e-health application functions f_1, f_2 and a unique identifier i_C. It is also copied with the identity of the e-health Server application i_S; these functions and parameters are used for mutual authentication and derivation of keys for subsequent session management. After the bytecode is downloaded to the card, the Card Manager uses the Global Platform specified Data Authentication Pattern to verify the integrity of the received ciphertext bytecode. Verification allows the ciphertext bytecode to be decrypted using K_{CI}. An applet instance is created and registered with the Java Card runtime environment.

8) $M \rightarrow C:$ APDU(`INSTALL`(Install): AID$_{SC}$)
The Applet is installed by the MIDlet issuing the `INSTALL` command to the Card Manager using the unique Applet application identifier AID_{SC}.

9) $M \rightarrow C:$ APDU(`INSTALL`(Selectabled): AID$_{SC}$)
This final instruction in the Applet installation phase instructs the Card Manager to allow the Applet to be selected by any application capable of issuing APDU commands. APDU communication with the Applet can now proceed under the control of the e-health provider's ME resident MIDlet application, which communicates directly with the e-health Server using any of the supported network protocols such as http or https.

This protocol phase uses standardised Global Platform [6] techniques to validate the origin and integrity of the Applet to ensure installation in the USIM.

PHASE 3: Perform mutual entity authentication

The security requirement for this protocol phase is to perform mutual entity authentication between the application provider Server and Applet without risk of a "man in the middle" or "replay" attacks. The choice of protocol to agree a session key is influenced by the constraints of the mobile environment: a MAC based approach limits network traffic while protection against replay attacks is through the use of nonces.

1) $S \rightarrow M:$ start MIDlet with push registry
If the Server side e-health application initiates the process to agree a sesion key then the MIDlet is invoke by J2ME push register.

2) $M \rightarrow C:$ APDU(`SELECT: AID`$_{SC}$)
The MIDlet commences a session with the e-health Server side application and selects the Applet application on the USIM using the unique application identifier AID$_{SC}$.

3) $C \rightarrow M:$ APDU(r_C)
Once invoked, the Applet generates a random nonce r_C, stores it, and supplies it to the MIDlet.

4) $M \rightarrow S:$ r_C
The MIDlet pases the nonce (without storing) back to the e-health Server.

5) $S \rightarrow M:$ $i_S \| i_C \| r_C \| r_S \|MAC_{K_{SC}}(\imath_S \| i_C \| r_C \| r_S)$
The Server generates a second nonce r_S, stores it together with the received nonce r_C and responds to the MIDlet with a challenge using the e-health shared secret K_{SC} and the Applet and Server identities i_C and i_S respectively.

6) $M \rightarrow C:$ APDU($i_S \| i_C \| r_C \| r_S \|MAC_{K_{SC}}(\imath_S \| i_C \| r_C \| r_S)$)
The MIDlet performs no function on the received string, but passes it onto the Applet in the form of an APDU command.

7) $C \rightarrow M:$ APDU($i_C \| r_S \|$MAC$_{K_{SC}}(i_C \| r_S)$)
After verifying that the received r_C and identifiers are correct, the Applet recalculates the MAC using the shared secret K_{SC} to authenticate the Server. To allow the Server to authenticate the Applet, the Applet provides the MIDlet with an integrity protected card identity i_C and Server nonce r_S as an APDU response
.

8) $M \rightarrow S:$ $i_C \| r_S \|$MAC$_{K_{SC}}(r_S \| r_C)$
Once the Server confirms the validity of the MAC and received values then the Applet is authenticated to the Server.

This protocol phase conforms to the three-pass mutual authentication protocol using MACs and nonces as specified in ISO/IEC 9798-4 clause 5.2.2 [10].

PHASE 4: Set up e-health session keys

The security requirement for this protocol phase is to derive session keys without risk of eavesdropping. Both Server and Applet contain identical functions $f1$ and $f2$ that are defined by, and known only to, the application provider. These functions are used to calculate the session cipher CK and integrity IK keys using the protocol nonces r_S and r_C and the long term shared secret K_{SC}.

$$CK = f1_{K_{SC}}(r_S \| r_C)$$
$$IK = f2_{K_{SC}}(r_S \| r_C)$$
$$K_S = CK \| IK$$

This key derivation technique using authentication nonces and a shared secret in an application specific function, conforms to international standards.

IV. Protocol Properties

The session key K_S is now shared between the application Server and the e-health Applet. The novel protocol presented in this paper provides the following privacy benefits over the 3GPP GBA protocols when applied to privacy sensitive applications such as e-health.

1) The mobile operator is not privy to the derived session keys K_S used to secure the application. The privacy performance of the Mobile Operator's operating procedures need not be considered eliminating the need to accredit the Mobile Operator as a trusted third party.
2) The MIDlet and Applet clients are both under the control of the application provider. There is no dependency, other than implementation of the java virtual machine, on the capability of native ME and USIM vendor code to resist attack.
3) The application vendor can determine the most appropriate client architecture for the needs of the application. Sensitive functions may be performed by the Applet within the tamper resistant USIM, whilst functions that require the processing power and I/O capabilities of the ME can be performed by the MIDlet. In sensitive applications the master session key K_s could remain within the tamper resistant USIM card, and a derived key provided to the MIDlet. Unlike GBA_U where the derived session key $K_S\_ext\_NAF$ is provided to the native GBA Boostrapping Client of the ME, with this novel approach, the derived key is provided directly to the application MIDlet, eliminating any dependency on native ME code.

From a deployment perspective, unlike GBA, this novel approach uses existing technology. Neither the ME nor the USIM need be GBA compliant. Furthermore it is not necessary for the participating mobile operator to provide a Bootstrapping Server Function service. The AKA protocol is also suitable for over-the-air deployment to pre-issued and capable, but unprepared, standard M.E.'s

V. Conclusion

This paper describes a novel alternative to the 3GPP GBA that could be deployed to incorporate the mobile end point within an e-health service.

Acknowledgment

This work was supported by sponsorship funding from Telefonica Móviles, España. The author wishes to thank the Information Security Group at Royal Holloway, University of London for their continued financial and academic support during the course of these studies.

References

[1] 3GPP TS 03.48. *Technical Specification Group Terminals; Security Mechanisms for the SIM application toolkit; stage 2.* http://www.3gpp.org, 2001.
[2] 3GPP TS 23.057. *Technical Specification Group Terminals; Mobile Execution Environment (MExE); Functional description; Stage 2.* http://www.3gpp.org, 2003.
[3] 3GPP TS 31.101. *Technical Specification Group Terminals; UICC-terminal interface; Physical and logical characteristics.* http://www.3gpp.org, 2003.
[4] 3GPP TS 31.102. *Technical Specification Group Terminals; Characteristics of the USIM application.* http://www.3gpp.org, 2003.
[5] ETSI TS 101 476. *Digital cellular telecommunication system (Phase 2+); Subscriber Identity Module Application Programming Interface (SIM API); SIM API for Java Card; Stage 2 (GSM 03.19).* ETSI, http://www.etsi.org, 2000.
[6] Global Platform. Open platform card specification 2.1, 2001.
[7] GSM 03.19, Version 8.2.0. *Digital Cellular Telecommunications System (Phase 2+); Subscriber Identity Module Application Programming Interface (SIM API); AIM API for Java Card; Stage 2.* ETSI, http://www.etsi.org, 2001.
[8] GSM 11.14. *Digital cellular telecomunnications system (Phase2+); Specification of the SIM Application Toolkit for the Subscriber Identity Module-Mobile Equipment (SIM-ME) interface.* ETSI, http://www.etsi.org, 2001.
[9] S. B. Guthery and M. J. Cronin. *Mobile Application Development with SMS & the SIM Toolkit.* McGraw-Hill, 2002.
[10] ISO/IEC 9798-4. *Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function.* International Organization for Standardization, http://www.iso.org, 2nd edition, 1999.
[11] JSR-118 JCP. *Mobile Information Device Profile, v2.0 (JSR-118).* Sun Microsystems, http://java.sun.com, 2002.
[12] JSR-177 JCP. *Security & Trust Services API (SATSA) (JSR-177).* Sun Microsystems, http://java.sun.com, 2004.
[13] R. Marti, J. Delgado, and X. Perramon. Security specification and implementation for mobile e-health services. In *EEE'04: Conference on e-technology, e-Commerce and e-Service*, pages 241–248. IEEE, 2004.
[14] Fred Piper and Sean Murphy. *Cryptography – A Very Short Introduction.* Oxford University Press, 1st edition, 2002.
[15] Wolfgang Rankl and Wolfgang Effing. *Smart Card Handbook.* John Wiley & Sons, Ltd, 3rd edition, 2003.
[16] K. Topley. *J2ME In a Nutshell.* O'Reilly, 2002.