Design, Installation & Execution of a Security Agent for Mobile Stations

William G. Sirett*, John A. MacDonald**, Keith Mayes, and Konstantinos Markantonakis

Smart Card Centre
Information Security Group
Royal Holloway, University of London,
Egham, England TW20 0EX
{w.g.sirett, k.markantonakis, keith.mayes}@rhul.ac.uk,
john@madgo.com

Abstract. In this paper we present a methodology and protocol for establishing a security context between a Mobile Operator's application server and a GSM/UMTS SIM card. The methodology assumes that the already issued Mobile Station is capable but unprepared. The proposed scheme creates a secure entity within the Mobile Station "Over The Air" (OTA). This secure entity can then be used for subsequent SIM authentications enabling m-Commerce, DRM or web service applications. To validate our proposal we have developed a proof of concept model to install and execute the security context using readily available J2ME, Java Card, J2SE and J2EE platforms, with the KToolBar MIDP2.0 emulator tool from Sun, and a Gemplus Java Card.
Keywords: Mobile Station, Security Agent, Application Deployment, Smart Card, GSM, Security Protocol, JSR177, MIDP2.0.

1 Introduction

The GSM network offers a wide scope of applications and benefits for mobile operators. The merits of a Mobile Station capable of implementing a *Security Agent* are well documented in the literature [17, 18]. In this paper we consider the deployment of a *Security Agent* that is comprised of two components: a device application executing resource-intensive tasks, and a secure entity application responsible for secure functionality. The secure entity is a tamper resistant [5] entity and in the case of this work is a GSM/UMTS SIM card. For some time the GSM network has allowed for "Over The Air" (OTA) SIM application installation with limited bandwidth capacity. To install these applications utilising a high bandwidth channel and a non-GSM specified protocol currently demands

* This work was supported by sponsorship funding from the Smart Card Centre founded by Vodafone and G&D.
** This work was supported by sponsorship funding from Telefonica Móviles, España.

trust/keys being provided to the Mobile Device. This work considers the Mobile Device to be hostile. This raises a need for the same high bandwidth OTA functionality to be available whilst protecting against malicious equipment.

This work establishes a security context between a Mobile Operator Application and Mobile Station and proposes an authenticated key establishment protocol. By establishing session keys independent of the network security keys, we can provide integrity, authentication and confidentially at the application layer. In the GSM/3GPP mobile architecture [24], the user security context resides in two locations, the network HLR and the Operator issued tamper resistant SIM card. The Mobile Operator generally has much less control over the Mobile device than the SIM. Consequently they are more reluctant to load sensitive components or data into the device. This motivates the division of the *Security Agent* between the device and the SIM, where the SIM is responsible for particularly sensitive components. We propose a scenario where the Mobile Operator Application server communicates with the device resident component of the *Security Agent*. This subsequently uses the security services provided by the secure entity to establish authenticated keys.

In section 2 we review the design requirements for a *Security Agent* deployed on a GSM/3GPP Mobile Station. In section 3 we review our proposed authenticated key establishment scheme. A protocol for wireless installation of the *Security Agent* to a compatible but remote and unprepared Mobile Station (colloquially termed "OTA" and "backward compatible field installation") is detailed in section 4, whilst the protocol used to establish session integrity and confidentiality keys is presented in section 5. Finally, in section 6, we describe the Proof of Concept model constructed using readily available components and open source development tool kits and provide concluding remarks in section 7.

2 Design Requirements

A critical requirement is for a backward compatible field installable *Security Agent* designed to provide an authentication service using SIM based credentials. It is required to be executable on a significant proportion of globally standardised and deployed Mobile Stations. The design of our proposed *Security Agent* uses four widely adopted technologies and standards:

- ETSI TS03.48 Security Mechanism [1];
- SIM Application Toolkit (SAT) [12];
- MIDP2.0 J2ME Runtime Environment [15];
- UICC Java Card SIM cards [3].

2.1 ETSI TS03.48 Security Mechanism

ETSI TS03.48 [1] specifies a mechanism for providing end to end security for any Short Message Service (SMS) going to or from the SIM card. SMS messages contain a maximum of 140 bytes. SMS messages are sent in accordance

with the SUBMIT_SMS format, and received with the SMS_DELIVER format. Translation from one format to the other is performed by the Short Message Service Centre (SMSC), an active component of the network. In an output SUBMIT_SMS packet, the 40 bytes of User Data are complemented by a 13 byte *Mandatory Header* and an optional variable length *User Data Header*.

- The *Mandatory Header* includes the Data Coding Scheme byte which specifies how the data is encoded, and the Protocol Identifier byte which specifies how the receiving mobile should process the message. One of these values, $0xF7$, specifies that the device should pass the whole packet to the SIM card.
- The *User Data Header* comprises a concatenation of tag, length and value (TLV) fields which describe the optional features that should be applied to the attached 140 bytes of user data. Of interest to our proposal are tag values $0x00$ and $0x70$, meaning Concatenated SMS and SAT Security respectively.
  - The concatenated SMS tag allows up to 255 SMSs to be concatenated. It is reported [13] that most operators limit this to approximately five, because of uncertain and indeterminate device operation when receiving larger numbers of SMS messages to be concatenated. Five messages represents a total payload of 5x140 = 700 bytes [18].
  - The presence of the SAT Security tag ($0x70$) indicates that the message contains an additional header, the *Command Header*, prior to the *User Data Header*. This comprises of 9 fields which define how the User Data is secured by:
    - specifying the cryptographic functions,
    - providing a replay protection counter,
    - quoting the sender's cryptographic integrity value for the secured *User Data Header*.

Through the use of this SAT Security mechanism it is possible to provide confidentiality and integrity services for up to 700 bytes of user data, when the data is sent between the Mobile Operator application server and the SIM card. Performance and payload are limited and applications are restricted.

2.2 SIM Application Toolkit

The SAT API allows an application on the SIM card to be informed of events by, and to issue commands to, the host mobile device. When an information flow is initiated to the SIM application, it is termed an event download, and when an information flow is initiated from the SIM application it is termed a proactive command. Using the proactive command SET_UP_EVENT_LIST, the SIM application can register to be informed of a number of events via the ISO/IEC 7816-4 ENVELOPE APDU command [8]. Of relevance to this paper is the SMS_PP or CELL_BROADCAST event, which downloads the contents of the received SMS to the SIM application as a compound TLV in the data field of an ENVELOPE APDU [13]. The SIM application's response to the ENVELOPE command is then returned to the sender in a response packet.

2.3 MIDP2.0 J2ME Runtime Environment

A Java application that runs on a Mobile Information Device Profile (MIDP) 2.0 device is known as a MIDlet and may be installed within a certain domain if it complies with the domain-specific access control requirements [6]. There are 4 domains specified for GSM compliant devices:

- Untrusted,
- Trusted 3rd Party,
- Mobile Operator,
- Manufacturer.

A Domain Protection Root Certificate (DPRC) controls MIDlet access to a domain. The DPRC must be made available at a specified location in the SIM application [15]. The MExE [2] security framework, when making an access control decision relies on signature verification of the signed MIDlet using the public key contained within the DPRC. Successful verification of the digital signature allows the MIDlet to be installed into the appropriate domain of the device.

Any MIDlet within a domain enjoys a set of unique permissions provided by that domain. The permission model allows these installed MIDlets access to restricted and sensitive APIs. The Security and Trust Services API [16] specifies that access to three of the four defined packages is limited to MIDlets located within the operator domain. These packages are:

- SATSA–APDU
- SATSA–JCRMI
- SATSA–PKI

These provide the ability for MIDlets to access trusted elements (i.e. a SIM card) using APDU communication, invoke a method of a remote Java Card object and provide support for digital signatures and credential management. It is worth nothing the value of being able to process cryptographic functions upon the device, as well as the smart card, as the card is a constricted environment [19].

2.4 UICC Java Card SIM Cards

The Global Platform specification [10] is the industry standard interface for downloading applications and is the most important international specification for application management in multi-application smart cards [11, 21]. This standard allows card issuers to securely manage third party applications independently of the operating system provider. Mobile Operator are increasingly deploying UICC [3] Java Cards, where the SIM application [4] is just one of the possible java applications [9] that the card is capable of running. Java applications that run on smart cards are known as Applets. A Card Manager is responsible for ensuring that new Applets are integrity checked and their source

authenticated prior to installation. This security service uses a secret key, K_{CI} that is embedded in the smart card prior to issue and termed the Card Issuer Key. Although a UICC SIM card can execute multiple Applets from different providers, the Card Manager application will be owned by the Mobile Operator who actually owns and issued the physical card.

3 The Proposed Scheme

Consider the scenario of having the requirement to remotely deploy our *Security Agent* to Mobile Stations in the field. The device is capable but unprepared; this section introduces the proposed scheme to install the *Security Agent*. This is again described as a protocol in section 4 and is represented in Fig. 1. The scheme is preparation for the execution protocol, detailed in section 5 that establishes session keys for future communication.

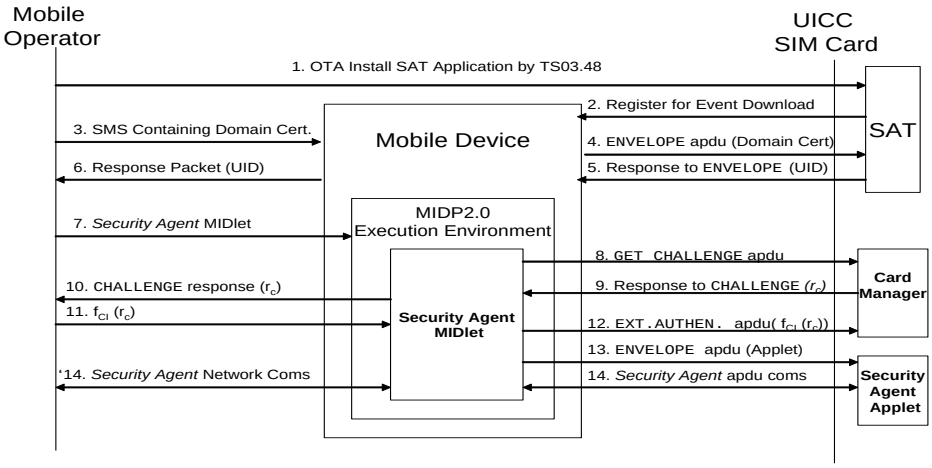

Fig. 1. Installation Scheme

1. A small SAT application is securely installed OTA to the SIM using the TS03.48 mechanism.
2. The SAT application uses the proactive command SET_UP_EVENT_LIST to register to be informed, via the ISO/IEC 7816-4 `ENVELOPE` APDU command [8], when a SMS_PP or CELL_BROADCAST event occurs.
3. A corresponding SMS_PP or CELL_BROADCAST is sent to the device. The DPRC is contained within the payload of a of the concatenated SMS messages.
4. The device transfers the payload to the SAT application as a compound TLV in the data field of an `ENVELOPE` APDU command. The DPRC is stored in the appropriate location of the UICC SIM Card [15].

5. The SAT application retrieves the SIM's unique identifier, and returns it to the device as the response to the `ENVELOPE` command.
6. The unique identifier is then returned to the Mobile Operator application server. This acts as a proof of delivery of the DPRC and enables the Mobile Operator application server to reference the card's secret key K_{CI} and commence the MIDlet preparation and download process.
7. Using the SIM's unique identifier, the Mobile Operator constructs the appropriate *Security Agent* MIDlet containing the relevant install commands and byte code for the *Security Agent* Applet. The Applet byte code is encrypted and digested with the card issuer key K_{CI} and packaged within the MIDlet JAR file.

   The MIDlet must be prepared for the secure MIDlet installation procedure defined by J2ME MIDP2.0 and implements a second security context between server and device. The server generates a RSA X.509(v3) certificate or requests one from a Certificate Authority (CA). The certificate is inserted into the application descriptor of the MIDlet application. The path of the descriptor holds all certificates necessary to validate the application except the root certificate. The DPRC resides on the smart card and is called into play during MIDlet installation. Finally, the signature of the JAR file (format used to distribute MIDlets) is generated with the private key of the RSA certificate according to the EMSA-PKCS-v1_5 encoding method of PCKS#1 version 2.0 standard. This signature is then inserted into the application descriptor, the MIDlet is considered prepared and delivered to the device.

   The J2ME JRE must authenticate the MIDlet application for installation into a secure domain. First the certificate is retrieved from the application descriptor and validated against the DPRC held upon the smart card. The JRE then verifies the MIDlet JAR file; by taking the public key from the verified signer certificate along with a fresh SHA-1 digest of the JAR file and comparing it to signature defined in the application descriptor. The JRE can install the MIDlet into the Operator domain of the MIDP2.0 runtime environment.
8. The *Security Agent* MIDlet is installed in the Operator domain of the user device with full access to JSR 177 APIs. This allows APDU commands to be issued to SIM card. The *Security Agent* MIDlet executes its Applet installation routine. The MIDlet starts by using the `SELECT` command to initiate communication with the SIM Card Manager. Once the Card Manager application is selected the MIDlet then issues a `GET CHALLENGE` command.
9. 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 the `GET CHALLENGE` with a random number r_C.
10. The MIDlet is not in possession of the secret K_{CI} required to prove a trusted source for a new Applet, so the challenge response r_C must be sent back to the Server.
11. The Server encrypts K_{CI} and r_C with K_{CI} and returns the byte string to the MIDlet.

12. The MIDlet authenticates the source of the applet using the `EXTERNAL AUTHENTICATE` command providing the encrypted response to the random challenge r_C. The Card Manager, also in possession of K_{CI}, authenticates the source of the Applet and allows the applet installation process to continue.
13. The *Security Agent* MIDlet now transfers the encrypted and integrity protected byte code of the *Security Agent* Applet to the SIM 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 SIM and Server for predefined parcels of bytes. The integrity and confidentiality of the applet code and the long term secret K_{CI} is assured. Subsequent to the downloading of the byte code to the card, the same Card Manager using, the Global Platform specified Data Authentication Pattern (DAP) verifies the integrity of the received byte code. Verification allows the byte code to be decrypted. A *Security Agent* applet instance is created and registered with the Java Card runtime environment.

    Contained within the *Security Agent* Applet byte code is a long term *Security Agent* symmetric key K_{SC} used for mutual authentication and establishment of secure session keys for subsequent execution of a *Security Agent* controlled authenticated key establishment process (Section 5).
14. APDU communication between the MIDlet and Applet components of the *Security Agent* can now proceed under J2ME application control. Communication between the Application Server and the J2ME environment may use any of the supported network protocols such as http or https for security services.

4 Installation Protocol

Our protocol uses both symmetric and asymmetric cryptographic techniques [20] to provide the authentication and integrity services required. The specific algorithms involved are either defined by the standards or rely on what the individual smart card supports.

Throughout this discussion we will use the following notation:

$$S = \text{Server}$$
$$M = \text{MIDlet}$$
$$C = \text{SIM card}$$

where:

K_{CI} = Shared secret between Server and SIM pre-issuance
K_{SC} = Shared secret between Server and SIM post-installation
Cert_{DPRC} = Domain Protection Root Certification
PK = Public key of Cert_{DPRC}
$\epsilon_K(D)$ = Symmetric encryption of data D using key K
$S_K(D)$ = Signature computed on data D using key K
$\text{MAC}_K(D)$ = MAC computed on data D using secret key K
r_E = Random nonce generated by entity E (S, C or M)
i_E = Identifier of entity E (S, C or M)
CK = Cipher Key
IK = Integrity Key
APDU() = APDU command from MIDlet to SIM card
SAT() = GSM SAT communication mechanism
SMS() = SMS communication mechanism

PHASE 1 *Install the MIDlet into the Operator Domain.*

$$S \rightarrow C: \text{SAT}(\text{MAC\_}ID) \quad (1)$$
$$S \rightarrow C: \text{SAT}(\text{SAT Applet Install code}) \quad (2)$$
$$S \rightarrow M: \text{SMS}(\text{Cert}_{DPRC}) \quad (3)$$
$$M \rightarrow C: \text{APDU}(\texttt{ENVELOPE}: \text{Cert}_{DPRC}) \quad (4)$$
$$C \rightarrow M: \text{APDU}(\texttt{ENVELOPE}: \text{UID}) \quad (5)$$
$$M \rightarrow S: \text{SMS}(\text{UID}) \quad (6)$$
$$S \rightarrow M: S_{PK}(\text{MIDlet}) \| (\text{MIDlet}) \quad (7)$$

Our protocol has been designed on the assumption that the device and SIM card are preissued and in the field, and although they are both capable, neither are prepared nor contain preinstalled application code to create the desired secure high bandwidth channel. The first step is to therefore prepare the SIM card so that the MIDlet can be installed within the Operator domain of the J2ME device. It is assumed, however, that the device is operational on the Operators network (i.e. user authentication and sign on to the network has successfully been performed by the `AUTHENTICATE` [4] and subsequent functions of the 3GPP challenge response mechanism [24]). Although messages (1) through to (3) are unidirectional from the Server to the SIM card or mobile, we have presented them as 3 individual protocol messages. This is because they are transferred using GSM standard 03.48 and are most likely sent as 3 independent SMS messages. Protocol messages (1) provides the SIM card with the identifier specifying

which GSM standard 03.48 MAC algorithm will be used to confirm integrity and data origin throughout the protocol sequence. Message (2) provides the SAT code be installed upon the card, whilst message (3) provides the root certificate of the device J2ME Operator domain. The payload of these messages are stored in the SIM card. The role of the SAT code is to discover the card's unique identifier (UID). By the means of the same mechanism the UID of the SIM card and r_C is securely sent back to the Server. This information is used to find the related K_{CI}, this key is used to encrypt and sign the *Security Agent* Applet byte code that is embedded and readied for delivery as part of the integral code that comprises the MIDlet.

Upon receipt of protocol message (7), the MExE [2] J2ME implementation on the client will verify the signature using the root certificate on Cert_{DPRC} previously stored into the SIM card via message (3). Valid verification provides data origin authentication and integrity of the MIDlet JAD and JAR files received. The *Security Agent* MIDlet is now installed OTA in the Operator domain of the J2ME MIDP2.0 compliant implementation of the client device, with full permissions to utilise SATSA-APDU and SATSA-PKI packages defined by JSR 177.

PHASE 2 *Install the Applet into the SIM card*

$$M \rightarrow C: \ \text{APDU}(\texttt{SELECT}\text{: AID}_{CM}) \tag{8}$$
$$M \rightarrow C: \ \text{APDU}(\texttt{GET CHALLENGE}) \tag{9}$$
$$C \rightarrow M: \ \text{APDU}(r_C) \tag{10}$$
$$M \rightarrow S: \ \text{SMS}(r_C) \tag{11}$$
$$S \rightarrow M: \ \text{SMS}(\epsilon_{K_{CI}}(K_{CI} \| r_C)) \tag{12}$$
$$M \rightarrow C: \ \text{APDU}(\texttt{EXTERNAL AUTHENTICATE}\text{: } \epsilon_{K_{CI}}(K_{CI} \| r_C) \tag{13}$$
$$M \rightarrow C: \ \text{APDU}(\text{MAC}_{K_{CI}}(\text{Applet}) \| \epsilon_{K_{CI}}(\text{APPLET})) \tag{14}$$
$$M \rightarrow C: \ \text{APDU}(\texttt{INSTALL}\text{(Install): AID}_{SA}) \tag{15}$$
$$M \rightarrow C: \ \text{APDU}(\texttt{INSTALL}\text{(Selectable): AID}_{SA}) \tag{16}$$

The MIDlet, now securely stored OTA in the device, carried an array of byte codes (see appendix 2). These byte codes represent an encrypted and signed CAP file from the Server using the shared secret between the SIM and Server, K_{CI}. The process of verifying the MIDlet during download and installation, using Cert_{DPRC}, has already verified the integrity of these byte codes as well as the MIDlet application. The steps (8) to (16) outline the process going on between device and card but the untrusted device would have no knowledge of what is being sent as it is protected by a secret that it is not privy to.

Step (8) sends the `SELECT` command to the card to communicate with the Card Manager on the SIM operating system that will handle the authentication of the card acceptance device. This authentication process is represented by steps (9) to (13), it begin with collecting a challenge, random number, from the card

using the `GET CHALLENGE` commands. This challenge is packaged in an SMS, step (11), and sent back to the Server. The Server using its shared secret K_{CI}, encrypts the number along with the key itself and sends it back to the MIDlet (12). The final authentication is performed by the Global Platform `EXTERNAL AUTHENTICATE` command which holds the encrypted challenge response (13). AID_{SA} refers to the unique application identifier of the *Security Agent* Applet. Once authenticated, steps (14) through (16) show the encrypted and signed download of *Security Agent* Applet to card, it's subsequent installation and final completion of the process allowing it to be selected by any Operator Domain located MIDlets.

The MIDlet pseudo code to generate this secure authentication, download and installation of *Security Agent* Applet to SIM from an untrusted device is presented in appendix 1.

5 Execution Protocol

At some time later, i.e. after the http session of PHASE 2 has closed, the Operator may choose to download bulk data securely from the Server to the SIM card. Before this can begin both endpoints must verify the identity of the other with a mutual entity authentication protocol. We take our authentication protocol from the ISO/IEC 9798 standard [14], deriving session keys for data origin authentication, data integrity and data confidentiality as part of our authenticated key establishment protocol. The choice of protocol is heavily influenced by the parameters and characteristics of our mobile environment. For similar reasons, as stated previously, authentication via symmetric cryptography is preferred, and a MAC based approach limits the amount of network traffic required to a minimum. The choice is further restricted owing to the time-less nature of the SIM card [21]. Only three possible sources of time are available; an internal clock, a remote server or a neighbouring device [22] and there are no other alternatives. To have an internal time keeper would require at least a portion of the card chip to have a source of permanent power [7]. Although it is, of course, possible for the SIM card to obtain a measure of time from the client device via the TIMER MANAGEMENT proactive command and TIMER EXPIRATION event download of the SAT API. The client device is, as stated previously, likely to be untrusted by Operators for any function concerning the communication of potentially network critical information, and cannot be used as a source of time for confirming message freshness. In consequence message timeliness, to protect against replay attacks, must be achieved with either logical time stamps or nounces. In conclusion, therefore we have adopted the three-pass mutual authentication protocol using MACs and nounces as specified in ISO/IEC 9798-4 clause 5.2.2.

PHASE 3 *Perform mutual entity authentication*

$$S \rightarrow M: \text{ start MIDlet with push registry} \quad (17)$$
$$M \rightarrow C: \text{ APDU(select Applet)} \quad (18)$$
$$C \rightarrow M: \text{ APDU}(r_C) \quad (19)$$
$$M \rightarrow S: r_C \quad (20)$$
$$S \rightarrow M: i_S \| i_C \| r_C || r_S \| \text{MAC}_{K_{SC}}(i_S \| i_C \| r_C || r_S) \quad (21)$$
$$M \rightarrow C: \text{ APDU}(i_S \| i_C \| r_C || r_S \| \text{MAC}_{K_{SC}}(i_S \| i_C \| r_C || r_S)) \quad (22)$$
$$C \rightarrow M: \text{ APDU}(i_C \| r_S \| \text{MAC}_{K_{SC}}(i_C \| r_S)) \quad (23)$$
$$M \rightarrow S: i_C \| r_S \| \text{MAC}_{K_{SC}}(r_S \| r_C) \quad (24)$$

Once again this step starts with an invocation of the push registry via message (17) and the device MIDlet *Security Agent* loaded in (7) selecting the SIM Applet loaded following message (16). The SIM card Applet generates a random nounce r_C, stores it, and supplies it to the MIDlet (19) where it is passed on (without storing) to the Server (20). Server generates nounce r_S, stores it together with received nounce r_C and responds with (21). Again this is passed through the MIDlet to the SIM card Applet via an APDU, message (22). Upon receipt the SIM card Applet verifies that the received r_C is the same as the one sent in (19) and that the identifiers are correct (note the UID could be used for i_C and a parameter of certificate Cert_{DPRC} supplied to the SIM card via message (3) used for i_S). The SIM card Applet then recalculates the MAC and if correct accepts the Server. Now that the Server is authenticated to the SIM card Applet, then Applet responds with message (23) via the MIDlet, which again passes it straight through to the Server in message (24). When Server receives message (24) it checks that the received value r_S is indeed the one sent in (21) and that the SIM card identifier i_B is correct. After confirming the MAC calculation the Server can then accept the SIM card Applet as valid. Following the mutual entity authentication of STEP 3, both Server and SIM card will establish Integrity IK and Confidentiality CK keys to protect the subsequent bulk data exchange between the Server and the SIM Card.

PHASE 4 *Set up session keys to protect the content to be downloaded*

$$CK = f1_{K_{SC}}(r_S \| r_C) \quad (25)$$
$$IK = f2_{K_{SC}}(r_S \| r_C) \quad (26)$$

Both Server and SIM card Applet will contain identical functions $f1$ and $f2$ to calculate the session cipher and integrity keys using the protocol nounces r_S and r_C and the long term shared secret K_{SC}. Now that session keys have been established the bulk data may be transferred to the SIM card encrypted for confidentiality with CK and concatenated with a MAC using IK for data origin authentication and integrity as necessary for the data being transferred.

6 Proof of Concept Model

To validate our proposal we constructed a proof of concept model, based on readily available open source tools; it comprised of:

– *Server:*
  A J2EE Servlet web application performed the Mobile Operator function and was packaged as a Web Application Archive (WAR) file for easy deployment on a Tomcat Apache Web Server.
– *Mobile Device:*
  The J2ME Client was emulated by the Wireless KToolbar [23] from Sun Microsystems and run our *Security Agent* MIDP 2.0 MIDlet on the reference J2ME implementation.
– *SIM card:*
  The SIM card *Security Agent* function was provided by a Gemplus GemXpresso smart card. This was a Java Card and adhered to a number of industry standards such as Global Platform and had on-card cryptographic capability. The equipment used to connect the card to the test-bed was a USB card reader and Gemplus RAD3 development environment was used in early tests to load *Security Agent* Applets. This could have been just as easily realised using a variety of different platforms, notably G&D development hardware and tools.

The demonstration environment for our model was implemented in J2SE. J2SE provides the necessary Java Swing classes for monitoring the various use case applications being tested. The model is designed so that each phase of a specific use case is initiated manually and monitored by visual feedback through the use of J2SE's GUI `LayoutManager` class and `ActionListener` interface.

There are different technologies involved in this scenario and as such only some aspects of the system could be placed within the scope of the practical work undertaken. The SAT application download process is well document and involves access to SMS generation so this was omitted. The MIDlet could be constructed and the communication between the server and MIDlet over a secure channel was considered within the scope of this work and practically demonstrated using a secure HTTP link between a web server and mobile device. Both entities employed Java based technologies and demonstrated a secure channel based upon mutual authentication with a shared secret. The cryptographic functionality involved could not be performed by the MIDlet as the SATSA-PKI and SATSA-CRYPTO packages are provided by the JSR-177 and at the time of investigation was unavailable. Our approach was to separate out this functionality to a J2SE application that would communicate with the MIDlet and in turn generate APDU commands to the smart card whilst perform any cryptographic functions required. This allowed the JSR-177 to be effectively modelled with only a moderate increase in complexity. The J2SE JSR-177 proxy used the Open Card Framework (OCF) to create a connection to the smart card and build command and response APDU for exchanges in data.

7 Conclusion

In this work we introduce a methodology, discuss the component technologies and define a protocol for establishing a security context between a Mobile Operator

application and SIM card. A solution is proposed to establish a *Security Agent* on an untrusted mobile device and trusted SIM card using "Over The Air" (OTA) techniques. A proof of concept demonstration capability is implemented with example java source code presented.

This work creates confidentiality and integrity session keys, CK and IK respectively. These are independent off all network security keys and therefore eligible to protect unrelated data, e.g. value added applications. The extension of trust from inherent network related credentials, to independent credentials used solely for providing security services to post-purchase applications, is an important step towards fulfilling the true potential of the mobile station field base by executing a new generation of secure applications.

8 Acknowledgements

We would like to extend appreciation to Chris Mitchell for early guidance and participation. Additionally, thanks to Jennifer Squire for patience whilst proof reading drafts.

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. R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *The Second USENIX Workshop on Electronic Commerce Proceedings, Oakland, California*, pages 1–11. USENIX Association, November 1996. http://citeseer.ist.psu.edu/400120.html.
6. C. Block and A. C. Wagner. *MIDP 2.0 Style Guide.* The Java Series. Addison-Wesley, London, 2003.
7. V. Cordonnier, A. Watson, and S. Nemchenko. Time as an aid to improving security in smart cards. In *7th Annual Working Conference on Information Security Management and Small Systems Security*, pages 131–144. Kluwer Academic Press, London, 1999. Amsterdam, The Netherlands.
8. ETSI TS 100 977. *Digital cellular telecommunications system(Phase 2+); Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface.* ETSI, http://www.etsi.org, 2000.
9. 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.
10. Global Platform. *Card Specification v2.1.1.* http://www.globalplatform.org, 2003.
11. 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.

12. 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.
13. S. B. Guthery and M. J. Cronin. *Mobile Application Development with SMS & the SIM Toolkit; Building Smart Phone Applications.* McGraw-Hill, 2002.
14. ISO/IEC 9798-4. *Information technology - Security Techniques - Entity Authentication - Part 4: Mechanisms using a cryptographic check function 2nd ed.,.* http://www.iso.org, 2nd edition, 1999.
15. JSR-118 JCP. *Mobile Information Device Profile, v2.0 (JSR-118).* Sun Microsystems, http://java.sun.com, 2002.
16. JSR-177 JCP. *Security & Trust Services API (SATSA) (JSR-177).* Sun Microsystems, http://java.sun.com, 2004.
17. J. A. MacDonald and C. J. Mitchell. Using the GSM/UMTS SIM to secure web services. In *2nd IEEE International Workshop on Mobile Commerce & Services (WMCS).* IEEE, IEEE Computer Society Press, July 2005. Munich, Germany.
18. J. A. MacDonald, W. G. Sirett, and C. J. Mitchell. Overcoming channel bandwidth constraints in secure SIM applications. In R. Sasaki, S. Qing, E. Okamoto, and H. Yoshiura, editors, *20th IFIP International Information Security Conference (SEC 2005) - Small Systems Security and Smart cards,*, volume 181 of *IFIP International Federation for Information Processing.* Springer Science and Business Media, May 2005. Chiba, Japan.
19. K. Markantonakis. Is the performance of the cryptographic functions the real bottleneck? In M. Dupuy and P. Paradinas, editors, *Trusted Information: The New Decade Challenge*, IFIP TC11 16th International Conference on Information Security (IFIP/SEC'01) June 11-13, pages 77–92. Kluwer Academic Publishers, 2001. Paris, France.
20. F. Piper and S. Murphy. *Cryptography - A Very Short Introduction.* Oxford University Press, 2002.
21. W. Rankl and W. Effing. *Smart Card Handbook.* John Wiley & Sons, Ltd, 3rd edition, 2003.
22. L. Rousseau. Secure time in a portable device. *Proceedings of 3rd Gemplus Developer Conference, Paris, France*, 2001. Gemplus.
23. Sun Wireless Toolkit. *Wireless Toolkit, Version 2.1,.* Sun Microsystems, http://java.sun.com/products/j2mewtoolkit, 2004.
24. M. Walker and T. Wright. *GSM and UMTS : The creation of global mobile communications.* John Wiley & Sons, Ltd., 2002.

A Appendix 1 - Psuedo Java Code

This pseudo code is intended to be a demonstration of the proposed method of embedding encrypted *Security Agent* Applet code within the body of the *Security Agent* MIDlet. The MIDlet can call the Install method of a packaged class called Secure_Applet and undertake the selection of the *Card Manager*, authentication of shared secret, downloading of CAP file, its installation and finalisation without actually exposing the underlying Operator secret to the MIDlet. The *Security Agent* Applet is encrypted within the MIDlet and its integrity is checked on-card during installation.

The following code is not intended to be a literal but uses Java based concepts to express the intention of the proposal. The aim is to illustrate the declaration of

a two dimensional array; a primary array holding arrays of 255 bytes representing APDU commands. The concept is that these APDUs can be declared by the Server during preparation of the MIDlet and therefore the device would not have the opportunity to alter or access the information. The only function the device or MIDlet must perform is the act of sending the APDUs to the device.

Listing 1.1. Psuedo MIDlet code

```
1  public class Secure_Applet{
2    //-- ISO7816 offsets
3    private bINS  = ISO7816.OFFSET_INS;
4    private bCLA  = ISO7816.OFFSET_CLA;
5    private bP1   = ISO7816.OFFSET_P1;
6    private bP2   = ISO7816.OFFSET_P2
7    private bLC   = ISO1716.OFFSET_LC;
8    private bData = IS07816.OFFSET_CDATA;
9
10   //--array of APDU byte codes
11   private byte[9][255] baAC
12       //-- select AID
13       = {00 A4 04 00 07 A0 00 00 00 18 43 4D},
14
15       //-- External Authenticate
16        {84 82 03 00 10 40 2F 82 CE 30 2C F5 78 F7 F7 60 32 0B 5A 4F 0E},
17
18       //-- First APDU of CAP file load
19       {84 E6 02 00 20 39 B0 DB 15 04 8D 75 BC 8D 71 46 83 52 A8 E2 D2
20       7D 48 32 25 AD DF DC 44 E2 28 55 2D 83 31 8B 34 00},
21
22       //-- following data packets
23        {84 E8 00 00 D8 B0 23 9E 36 52 BF 40 03 A1 F1 43 D8 3D 6A F8 93
24        //--consider full 255 byte apdu data array
25        71 0C 6D B3 41 56 B8 09 84 71 7C},
26        {...},{...},{...},{...},{...},{...};
27
28   public installSecureApplet() {
29     try {
30       //-- create card service and connect
31       //-- send all APDU commands confirming
32       //-- for length of parent array  loop
33       for (int x=0; x<baAB.length; x++){
34             //-- build APDU object using array and offsets
35             cmdAPDU = new ISOCommandAPDU(
36                 baAC[x][bINS], baAC[x][bCLA], baAC[x][bP1],
37         baAC[x][bP2], baAC[x][bData]);
38             //-- send APDU and catch response.
39             resAPDU = service.sendCommandAPDU(cmdAPDU);
40             if(Integer.toHexString(resAPDU.sw()) != "90_00"){
41                 break:}}}
42      //--catch errors and close down all objects}}
```