Coopetitive Architecture to Support a Dynamic and Scalable NFC Based Mobile Services Architecture

Raja Naeem Akram[1,2], Konstantinos Markantonakis[1], and Keith Mayes[1]

[1] ISG Smart Card Centre, Royal Holloway, University of London Egham, Surrey, United Kingdom
[2] School of Computing, Edinburgh Napier University, Edinburgh, United Kingdom
`R.Akram@napier.ac.uk`,
`{K.Markantonakis,Keith.Mayes}@rhul.ac.uk`

Abstract. Near Field Communication (NFC) has reinvigorated the multi-application smart card initiative. The NFC trials are relying on an extension of Issuer Centric Smart Card Model (ICOM) referred as Trusted Service Manager (TSM) architecture, which may create market segregation. Where the User Centric Smart Card Ownership Model (UCOM) takes an opposite approach of delegating the smart card ownership to its users. Therefore, to reconcile these two approaches we proposed the Coopetitive Architecture for Smart Cards (CASC) that avoids market segregation, increase revenue generation, and provide flexibility, robustness, and scalability. To support the CASC framework in this paper, we propose an application installation protocol that provides entity authentication, trust assurance and validation, mutual key and contractual-agreement generation. The protocol is compared with existing protocols on its performance, stated security, and operational goals. Furthermore, CasperFDR is used to provide a mechanical formal analysis of the protocol.

1 Introduction

In late 1990s, the multi-application smart card initiative enabled heterogeneous applications to co-exist and share resources in a secure and reliable manner [1]. At the time, it was envisioned that diverse organisations would converge with their services on a single device [2]; however, the reality has been different.

The issues related to the card ownership, marketing potential of the card surface, customer loyalty, and potential revenue stream, hindered any meaningful collaboration effort [3]. In addition, there were other voices mainly concerned with the security implication [4]. The enthusiasm died quickly until a new technology termed as Near Field Communication (NFC) emerged that enables a mobile phone to emulate a contact-less smart card [5]. Since 2007, NFC based mobile services with applications like banking, telecom, and transports are in trial around 38 countries [6]. In these trials, the smart card management architecture is based on the framework that has been deployed in the smart card industry since its inception, namely Issuer Centric Smart Card Ownership Model (ICOM). In the ICOM, smart cards are issued and controlled by a centralised authority known as a card issuer. Application providers require prior-authorisation from the card issuers to install their applications. The extension of the ICOM deployed in the NFC based trials is termed as Trusted Service Manager (TSM) architecture [7]. The TSM is an entity that can be either a card issuer or an independent third party. It manages the card platform, and relationship with individual stakeholders.

In contrast, User Centric Smart Card Ownership Model (UCOM) [3] is based on the citizen ownership architecture. In this model, cardholders (users) own smart cards, and they have the choice to install or delete any application. To reconcile between the UCOM and TSM, we proposed the Coopetitive Architecture for Smart Card (CASC)[1] that merges the TSM and UCOM frameworks, thus increasing the overall scalability of the multi-application smart card architecture, and possibly provide more revenue-generating opportunities than the TSM can individually achieve.

1.1 Contributions

In this paper, based on the CASC architecture, we propose a trusted and secure entity authentication, key generation, and contractual-agreement protocol for application download referred as Application Acquisition and Contractual Agreement Protocol (ACAP). The contractual-agreement guarantees to the participating entities that they have executed the protocol and as a successful outcome, an application is installed (and the application is operational).

1.2 Organisation

In section two, we provide a brief motivation behind the coopetitive architecture. A succinct discussion on the smart card architecture that supports the CASC framework is provided in section three. In this section we only discuss elements of the smart card design that is required to support the proposed protocol. These two sections set the background on which we base the security and operational requirements of the proposed protocol. Next in section four, the description of the ACAP is provided. Section five analyse the ACAP to see whether it meets the stated goals and requirements in comparison to existing protocols. In addition, we discuss the implementation experience and performance measurement of the ACAP along with formal analysis based on the CasperFDR. Finally, in section six we provide concluding remarks and list future research directions.

2 Motivation for Coopetitive Architecture

The TSM architecture, in a simplistic form, is illustrated in figure 1. In such an environment, a customer of a Mobile Network Operator (MNO) that has a

[1] To facilitate the blind reviewing process, references to CASC are removed.

relationship with the TSM-1 will only be able to have applications from Card Issuing Bank (CIB), Transport Service Provider (TSO) and leisure centres that are associated with the TSM-1. However, if the respective customer C_A does banking with the CIB_2 that is associated with the TSM-2 (figure 1) then either she has to acquire a new smart card from the TSM-2 or change bank. Therefore, in such a scheme, there is a potential for the segmentation of the market.

One possible option is to have all application providers maintain relationships with all or most of the TSMs. For example, in figure 1, the CIB_1 of TSM-1 should also have a relationship with the TSM-2. Another possible option is to create a syndicated scheme in which multiple TSMs participate.

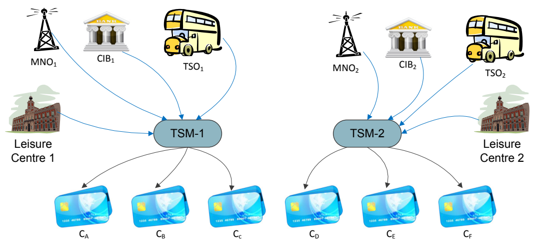

Fig. 1. Trusted Service Manager Architecture

Therefore, any application provider affiliated with one TSM will be able to issue its application to a customer of any syndicated TSM. Both scenarios can be argued to be workable, but they also suffer from limited scalability, flexibility, and ubiquity of the framework.

The limited scalability roots from: (a) not all application providers could establish or manage relationships with every possible TSM, and (b) not all TSMs would be part of a single syndicated TSM. In addition, to be part of a collaborative scheme a TSM might require subscription fee from application providers. Therefore, small or medium-scale organisations like local libraries, universities, and health centres, etc., may not be able to afford it. We consider that such a barrier to enter the scheme reduces its flexibility. Furthermore, it lacks true ubiquity as different countries might opt for having their own independent TSMs. Thereby, tourists or business travellers would face difficulty in acquiring applications (e.g. TSO's application) in a foreign country. These issues are on top of the ones that are discussed in [4] including ownership privileges, customer loyalty and relationship management, card surface marketing, and revenue generation [3].

In the UCOM, most of the issues discussed above are not present [3]. We consider that UCOM itself will be a preferable solution, but it is difficult to conceive that it can have a widespread acceptance in the business community. Therefore, a compromise between the TSM and UCOM is referred as Coopetitive Architecture for Smart Cards (CASC).

The coopetitive architecture focuses on the core competences of individual companies and leaves other areas to the organisations that have expertise in them. For example, an MNO in the coopetitive architecture can be a TSM and even have the ability to form alliance with other companies to provide their services via the respective smart cards. In addition, it also enables the users to download applications they like from any of the application providers of their choice. The main stake the MNO has is to generate maximum revenue out of its investment in the secure element, and its security. Therefore, if there is a way in which an application can be securely downloaded onto a smart card that does not have any prior relationship with the particular application provider and the MNO charges the customer for acquiring the application. Then in such a model, there is a probability that customers would actually generate higher revenue for the respective card issuer or TSM than in the traditional architecture based on the ICOM.

3 Coopetitive Architecture for Smart Cards

In this section, first we discuss the coopetitive architecture and then briefly describe the multi-application smart card architecture to support it.

3.1 Smart Card Architecture Overview

A generic architecture is illustrated in figure 2, for brevity we will only discuss those components that are related to this paper. On top of the hardware layer is the Trusted Environment & Execution Manager (TEM), which is discussed in the next section.

Above TEM is the smart card runtime environment that might conform to any of the smart card platforms or operating systems (e.g. Java Card [8] or Multos [9]). The smart card firewall manages the inter-application communication and access to the platform services (i.e. APIs). The top most layer is partitioned into three sections separated by the firewall mechanism: namely the Platform's, TSM's, and Cardholder's space. The Platform's space holds the platform APIs, where application related to individual entities (e.g. TSM and cardholder) are in their respective spaces.

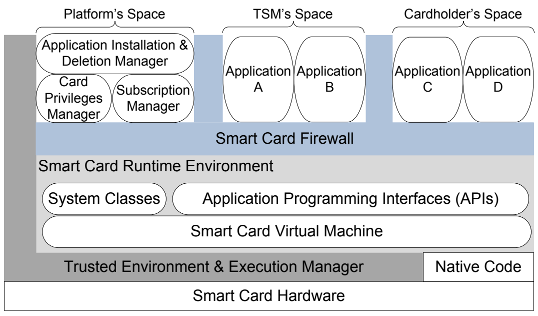

Fig. 2. Generic Smart Card Architecture for Coopetitive Framework

3.2 Trusted Environment and Execution Manager (TEM)

A TEM provides a platform independent dynamic, runtime, and remote – security and reliability assurance mechanism for the UCOM based smart cards. In a naive manner, we can term it as a trusted platform base for the smart cards; however, TEM's functionality differs from the traditional Trusted Platform Module [10]. For the sake of concision, we will only discuss the TEM component referred as the attestation handler in this section that is directly related to this paper.

The attestation handler implements the security-assurance and validation mechanism that certify to the requesting entity (e.g. Service Provider: SP) that the smart card's state is as it was at the time of a third party evaluation and stated by the evaluation certificate [11]. An evaluation certificate is a cryptographically signed certificate issued by an evaluation body, and the respective card manufacturer places it on the platform. The evaluation certificate will certify a unique signature key pair of the Smart Card Manufacturer (SCM). The SCM will use the signature key to issue certificates to the manufactured smart cards that conform to the evaluated product (see figure 3). A point to note is that at present Common Criteria (CC) [12] or any other evaluation scheme, for that matter, does not provide any such service but proposals presented in [11] and [13] can be utilised.

The process initiated by the attestation handler validates both the hardware and software state of the platform. It is a two-part mechanism: tamper-evidence and reliability assurance. To make a smart card tamper-resistant, the respective SCM implements hardware based tamper protections. The tamper-evidence process verifies whether the implemented tamper-resistant mechanisms are still in place and effective. The reliability assurance process verifies that the software part of the smart card platform is not been tampered/modified. For the description of the TEM and implementation of the attestation handler see [14].

4 Application Acquisition and Contractual Agreement Framework

In this section, we detail the security and operational goals for the Application Acquisition and Contractual Agreement protocol (ACAP) that facilitates application installation/deletion in the CASC, and propose a protocol that meets them.

4.1 Security and Operational Goals

An ACAP for the CASC should meet sixteen goals stated in [14] along with the additional goals listed as below:

G17) Platform & Application User Separation (PAU) Attack: A malicious user provides access credentials of a genuine user to an SP and downloads the application on her smart card [14]. A protocol should tie a platform with its respective card-owner (user) to avoid platform & application user separation attack.

G18) Contractual Agreement: On the successful execution of the protocol, the communicating entities will mutually sign a contractual agreement. This will act as a proof that a particular application was installed on a smart card.

G19) Proof of Transaction: The smart card will notify the TSM about the application installation. Depending upon the TSM's policy, it will charge the user's account and notify the smart card to activate the application so it can execute.

For formal definition of the italicised terms in the above list, readers are advised to refer to [15]. Later, we will revisit these goals for the protocol comparison (see table 3).

4.2 Enrolment Phase

A Smart Card Manufacturer (SCM) will get their smart card product certified from a certification authority that would issue a Product Evaluation Certificate (PEC), as shown in figure 3. It will endorse that the platform conforms to the stated security and operational requirements [14], along with the attestation process and its effectiveness.

The SCM may deliver the smart cards to a card issuer (e.g. a TSM) that will also certify the smart card signature key pair. Now, it will have two certificates, one issued by the SCM and second by the TSM. Finally, the respective smart card will be acquired by a cardholder who will then initiate the ownership acquisition process, which would generate a user signature key pair, certified by the smart card.

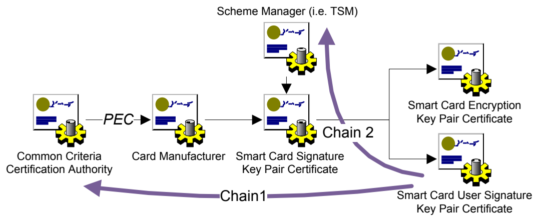
Scheme Manager (i.e. TSM)
PEC
Chain 2
Smart Card Encryption Key Pair Certificate
Common Criteria Certification Authority
Card Manufacturer
Smart Card Signature Key Pair Certificate
Chain1
Smart Card User Signature Key Pair Certificate

Fig. 3. Certificate Hierarchy in the Coopetitive Framework

There are two roots in this hierarchy (figure 3), the CC certificate authority, and the TSM. The reasons for having two separate roots are: a) to provide privacy protection to users who do not want to reveal the identity of their TSMs, and b) smart cards may not be permanently bonded with a particular TSM.

Depending upon the association of an SP with the TSM of a smart card, the appropriate chain of certificates will be provided by the smart card. If the SP is not an associate of the TSM, then the certificate chain 1 (figure 3) with the CC certification authority as a root will be used; otherwise, chain 2 will be used.

4.3 Proposed Protocol

Software on a mobile phone that supports the application installation process is referred as Card Application Management Software (CAMS) [16] in the UCOM. A cardholder requests the respective SP to download an application that initiates the ACAP protocol. The notation used to describe the ACAP is listed in table 1, where ACAP messages are listed in table 2 and discussed as below:

Message 1. The SP will initiate the ACAP by generating a random number (N_{SP}) and Diffie-Hellman exponential (g^{sp}) [19]. For computational efficiency, the SP might have a pre-computed buffer of random numbers and Diffie-Hellman exponentials. To avoid DoS attacks, the SP will compute a Session Identifier (SID) by $SID = H_{SP_k}(g^{sp}|N_{SP}|SC_{IP})$. The key ($H_{SP_k}$) used to generate the HMAC is not shared with any-other entity and SC_{IP} is the current Internet Protocol (IP) address of the respective smart card. When an SP will receive a

Table 1. Protocol Notation

SP	:Denotes a Service Provider.
SC	:Denotes a smart card.
T	:Denotes the enrolled TSM.
U	:Denotes a cardholder (user).
App	:Denotes the downloaded application contents.
X_i	:Represents the identity of an entity X.
g^x	:Current Diffie-Hellman exponential ($mod\, p$) generated by the entity X.
C_X	:Signature key pair certificate of an entity X.
N_X	:Random number generated by an entity X.
$A \rightarrow B$	:Message sent by an entity A to an entity B.
$X\|Y$	:Represents the concatenation of the data items X, Y in the given order.
$Sig_X(Z)$	:Is the signature on data Z by an entity X using a signature algo [17].
$H(Z)$	:Is the result of generating a hash of data Z.
$H_k(Z)$	:Is the result of generating a keyed hash (HMAC) of data Z using k.
$[M]^{eK_{X-Y}}_{aK_{X-Y}}$	:Message M encrypted by the encryption key eK_{X-Y} and then MAC is computed using the key aK_{X-Y}, shared between entities X and Y.
DH_G	:Details the Diffie-Hellman group that is used to generate the g^{SP} [18].
ALP	:SPs defines the Application Lease Policy (ALP) [16] that states the minimum security and operational requirements an SC has to meet to get the application lease. The application can be downloaded only after the SC satisfies the lease requirements [3].
$ReqV$	:The message sent by the respective SP to a SC requesting to provide a current state validation.
CAR	:List of cryptographic algorithms supported by the respective SP.
CAS	:List of cryptographic algorithms selected by the respective smart card from the CAR.
$ParOpt$	:Optional parameters of the protocol messages.
$AppDoD$	:An anonymised message that details the application properties (e.g. size) and it is used for charging purposes by the scheme manager.

message from the SC, it will first verify the SID. If the SID corresponds to an open session, and it computes correctly for the stated IP address (from where the message is received), then the SP will proceed with processing the message.

Message 2. On receipt, the SC will first check the DH_G whether support the selected group or not. If it cannot support the selected group then the smart card will sends a rejection message that lists the DH groups supported by the smart card. The SC then verifies whether it satisfy the ALP requirements. In addition to the security and operational requirements, the ALP also stipulates the required memory to install the application. The SC checks, whether it has enough available space to accommodate the requested application. If the SC cannot satisfy the ALP, it will terminate the protocol and notify the cardholder.

Otherwise, the SC will then generate a random number (N_{SC}) and Diffie-Hellman exponential (g^{sc}). It can now also generate the shared key (i.e. $DH = (g^{sp})^{sc}\ mod\ p$) and from this key, the SC will generate the session encryption $K_e = H_{DH}(N_{SP}|N_{SC}|0)$ and MAC key $K_a = H_{DH}(N_{SP}|N_{SC}|1)$. Session keys for the application download process can also be generated in the similar fashion.

Table 2. Application and Contractual Agreement Protocol (ACAP)

M1. $\mathcal{SP} \to \mathcal{SC} : N_{SP}|g^{sp}|DH_G|ALP|SID$

M2. $\mathcal{SC} \to \mathcal{SP} : N_{SC}|g^{sc}|[Sign_U(SC_i|U_i|g^{sp}|g^{sc}|N_{SP}|N_{SC}|PEC)|C_U]^{eK_{SC-SP}}_{aK_{SC-SP}}|SID$

M3. $\mathcal{SP} \to \mathcal{SC} : RV|\,[Sign_{SP}(SP_i|App_i|g^{sp}|g^{sc}|N_{SC}|N_{SP})|C_{SP}|CAR|PO]^{eK_{SC-SP}}_{aK_{SC-SP}}$

M4. $\mathcal{SC} \to \mathcal{SP} : [Sign_{SC}(SC_i|U_i|N_{SC}|N_{SP}|PO)|CAS|\,C_{SC}]^{eK_{SC-SP}}_{aK_{SC-SP}}|SID$

M5. $\mathcal{SC} \to \mathcal{SP} : [Sign_{SC}(H(App)|SP_i|App_i|ALP|SC_i|U_i|N_{SC}|N_{SP})]^{eK_{SC-SP}}_{aK_{SC-SP}}|SID$

M6. $\mathcal{SP} \to \mathcal{SC} : [Sign_{SP}(H(App)|SC_i|U_i|SP_i|N_{SC}|N_{SP}|PO)|\,C_{SP}]^{eK_{SC-SP}}_{aK_{SC-SP}}$

M7. $\mathcal{SC} \to \mathcal{T} : Card_{ID}|\,[T_i|SC_i|U_i|AppDoD|N'_{SC}]^{eK_{SC-T}}_{aK_{SC-T}}\,|SID_{T-SC}$

M8. $\mathcal{T} \to \mathcal{SC} : [Sign_T(T_i|SC_i|U_i|N_T|N'_{SC}|TC|ActApp)|Card'_{ID}|SID'_{T-SC}]^{eK_{SC-T}}_{aK_{SC-T}}$

The SC will sign the data containing the PEC (Product Evaluation Certificate) with user's signature key, then it is concatenated with the user's certificate. The entire message is encrypted and MACed, and appended to the g^{SC} and N_{SC}.

Message 3. The SP will retrieve the g^{sc} and calculate $DH = (g^{sc})^{sp}\ mod\, p$. Similar to the SC, the SP will also generate the session encryption and MAC keys.

The SP verifies the user's certificate, and the details of the cardholder listed in the user's certificate should match the SP's authenticated customer that requested the application download. This is to avoid users from installing applications for which they are not authorised (i.e. see requirement 17 in section 4.1). The SP verifies the signature and checks whether the PEC meets the minimum security-requirement set out in the SP's ALP. If it does not, the SP will terminate the protocol.

If there is no error, the SP will request (i.e. RV) the SC to provide a proof that it complies with the stated PEC. The SP then appends the encrypted message that contains cryptographic algorithms supported (CAR) by the SP (i.e. for use in application download), and an optional parameter (PO). The PO field is used by the SP if its application also has a third party evaluation certificate that is attached the certificate with message.

Message 4. On receipt of the message 3, the smart card verifies whether it supports the cryptographic algorithms listed in the CAR. If not, then the SC will send a list of cryptographic algorithms supported by the SC. If the SP does not support any of them, it can terminate the protocol and notify the user.

Otherwise, the SC will check whether the SP's identity is included in the associated SP's list (section 4.2). If it is included, the application will be installed in the TSM's space (section 3.1). If the SP's identity is in the list, then in the response message the SC will include the TSM's identity as an optional parameter (PO).

The SC will then initiate the platform attestation process as discussed in section 3.2. A correct signature that includes the protocol related data (i.e. random numbers and identities) will ascertain that the smart card is still in conformance with the evaluated state. The SC also includes the list of cryptographic algorithms (CAS) selected from the CAR list for the application download process.

Up to this point, the protocol achieves the entity authentication (e.g. SC, user, and SP), provides SC trust validation proof, and has generated session keys. For performance comparison (table 4), we refer to the message 1-4 as AKG (Authentication, Key Generation and Trust Validation) phase. After receiving the message 4, the respective SP will initiate the application download process, which is beyond the scope of this paper. However, for completeness, the communicating parties may choose one of the symmetric key-based application download protocol from the GlobalPlatform specification [20].

Message 5. Once the application download is completed, the SC will generate a message that acts as an SC to SP contract. The SC will generate the hash of the downloaded application and sign it with identities of the SP, downloaded application, SC, and the user.

Message 6. The SP verifies the $H(App)$ generated by the SC and activates the application lease to the user on the SP's server. The application lease activation does not mean that the respective SC can be used to access SP's services. The access to services is only activated at the successful conclusion of the ACAP, when the SC activates the application and it dials back home to activate the access to sanctioned services. Similarly, the SP's application is not activated on the SC; it is in the blocked (dormant) state. If the SP is not associated with the scheme TSM, then the SP will sign the message containing $H(App)$.

To activate an application, the SC requires the scheme TSM's authorisation. If the SP is associated with the TSM, it will send the identities of the SC, user, and downloaded application to the respective TSM. The TSM in reply will generate the $ActApp = Sign_T(App_i|SP_i|SC_i|Ui|N_{SP}|N_{SC})$. The $ActApp$ acts as an application activation message and it will be included in the message 6 as an optional parameter ($ParOp$). In this scenario, the last two messages will be redundant and will not be executed. This message acts as an SP to SC contract.

Message 7. In scenarios where the SP is not a member of the TSM, the user has to pay for the application download as per TSM policy and after this the TSM will issue the $ActApp$. The SC will request the TSM to issue $ActApp$ by sending the above message. The SC will use a one-time pseudo card identity ($Card_{ID}$) (i.e. privacy reason) so that an eavesdropper may not be able to match the $Card_{ID}$ uniquely to the SC_i. The SC will encrypt the message containing the identities of TSM, SC, and user. It then appends the application details ($AppDoD$) and a new random number generated by the SC. The $AppDoD$ will not have any details of the application that can help the TSM to uniquely identify either the SP or the application. It will include the memory occupied by the application, and if the TSM charges the user according to the space usage then this data will be used to calculate the charge. Finally, the SC uses the one-time SID_{T-SC} that is generated in previous protocol runs with the TSM, to avoid the DoS attack on the TSM's server; . The process to generate the $Card_{ID}$ and SID_{T-SC} is explained in the next message.

Message 8. The TSM will first verify whether the $Card_{ID}$ and SID_{T-SC} corresponds to the values in its database so that it will process the transaction and charge the user's account. Afterwards, the TSM will sign the message that includes the transaction certificate of the charge performed by the TSM and the *ActApp*. The *ActApp* is generated similarly as detailed in the message 6; however, the value in the App_i is a pseudo value that has no relation with the actual identity of the downloaded application. Finally, the TSM will generate the SID for the next session $SID'_{T-SC} = H_{K_T}(Card'_{ID}|T_i|SC_i)$ and $Card'_{ID} = H(T_i|SC_i|N'_{SC}|N_T)$. The TSM will store the generated $Card'_{ID}$ and SID'_{T-SC} in the internal database.

After the SC receives the *ActApp*, it activates the application and notifies the cardholder about the successful outcome of the application installation, and any incurred charge. The charging mechanism for the individual transactions is on sole discretion of the respective TSM.

5 Analysis of the ACAP Protocol

In this section, we analyse the proposed ACAP in terms of informal analysis, mechanical formal analysis (CasperFDR), and practical implementation with performance comparison.

5.1 Brief Informal Analysis of the Protocol

In this section, we constantly refer to the protocol requirements and goals for the ACAP; therefore, here onward any reference to a goal or requirement number refers to the listed item in section 4.1.

As shown in the table 3, the most promising results were from the ASPeCT and JFK protocols that meet a large set of goals. The T2LS protocol [28] meets the trust assurance goal by default, but similar to SCP81 it is based on the TLS protocol, which does not meet most of the requirements. A note in favour of the SCP10, SCP81, MM, and SM protocol is that they were designed with the assumption that an application provider has a prior trusted relationship with the smart card issuer; thus implicitly trusting the respective smart card. Whereas, the proposed ACAP protocol meets all the listed goals.

5.2 Protocol Verification by CasperFDR

The CasperFDR approach was adopted to test the soundness of the proposed protocol under the defined security properties. In this approach, the Casper compiler [29] takes a high-level description of the protocol, together with its security requirements. It then translates the description into the process algebra of Communicating Sequential Processes (CSP) [29]. The CSP description of the protocol can be machine-verified using the Failures-Divergence Refinement (FDR) model checker [29]. The intruder's capability modelled in the Casper script for the proposed protocol is as: 1) an intruder can masquerade any entity in the network,

Table 3. Protocol comparison on the basis of the stated goals (see section 4.1)

Gs	**Protocols**								
	STS[21]	AD[22]	ASPeCT[23]	JFK[24]	T2LS	SCP81[25]	MM[26]	SM[27]	ACAP
G01.	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$-*$	$*$
G02.	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
G03.	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
G04.	$*$	$*$	$*$	$*$	$*$	$*$			$*$
G05.	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
G06.	$*$		$*$	$*$			$*$	$-*$	$*$
G07.	$*$	$*$	$*$	$*$	$*$	$*$	$*$		$*$
G08.	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
G09.	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
G10.	$*$		$*$	$*$	$*$	$*$			$*$
G11.	$*$	$(*)$	$+*$	$*$	$*$	$*$	$+*$	$+*$	$*$
G12.	$(*)$	$(*)$		$(*)$	$(*)$	$(*)$			$*$
G13.					$*$	$-*$			$*$
G14.				$*$					$*$
G15.	$(*)$		$*$	$*$					$*$
G16.					$-*$				$*$
G17.									$*$
G18.									$*$
G19.			$*$		$+*$	$+*$	$+*$	$+*$	$*$

Note: $*$ means that the protocol meets the stated goal, $(*)$ shows that the protocol can be modified to satisfy the requirement, $+*$ shows that protocol can meet the stated goal but requires an additional pass or extra signature generation, and $-*$ means that the protocol (implicitly) meets the requirement not because of the protocol messages but because of the prior relationship between the communicating entities.

2) (s)he can read the messages transmitted by each entity in the network, and 3) (s)he cannot influence the internal process of an agent in the network.

The security specification for which the CasperFDR evaluates the network consists of: 1) the protocol run is fresh and both applications were alive, 2) the key generated by the SP and SC is not known to the intruder, 3) entities have mutually authentication and key assurance at the conclusion of the protocol, 4) long terms keys of communicating entities are not compromised, and 5) the user's identity is not revealed to the intruder. The CasperFDR tool evaluated the protocol and did not find any feasible attack(s).

5.3 Practical Implementation

The proposed ACAP does not provide any specific details of the cryptographic algorithms to be used during the protocol run. This choice is left to the respective SPs and smart cards. To provide a performance measure for the ACAP, we have used Advance Encryption Standard (AES) [30] 128-bit key symmetric encryption with Cipher Block Chaining (CBC) [15] without padding for both encryption and MAC operations. The signature algorithm is based on the Rivest-Shamir-Aldeman (RSA) [15] 512-bit key. We have used SHA-256 [31] for the

Table 4. Protocol Performance Measures (Milliseconds)

Protocols \Phases	SSL [32]	TLS [33]	Kerberos [34]	ACAP	
				Card One	Card Two
	32-bit	32-bit	32-bit	16-bit	16-bit
AKG Phase (M1-4)	4200	4300	4240	3395	3559
Contract Phase (M5-6)	-	-	-	1253	1294
Charge Phase (M7-8)	-	-	-	1407	1470
Total (M1-8)	-	-	-	6055	6323

hash generation by the TEM. For Diffie-Hellman key generation we used 2058-bit group with 256-bit prime order subgroup specified in the RFC-5114 [18].

The architecture of the ACAP test-bed is based upon three entities: a smart card, an SP and a TSM. The entities SP and TSM are implemented on a laptop with 1.83 GHz processor, and 2 GB of RAM, running on Windows XP. The smart card entity is implemented on a 16-bit Java Card and the implementation takes 9799 bytes of memory space. The implemented protocol was executed for 1000 iterations and time taken to complete individual iteration was recorded. The performance measures are taken from two different 16-bit Java Cards, and an average of recorded measurements for both cards is listed in table 4. For comparison, we have selected the SSL performance measured by Pascal Urien [32], TLS from Urien and Elrharbi [33], and (public key based) Kerberos by Harbitter and Menascé [34].

The rationale behind the choice of SSL and TLS for comparison lies in the GlobalPlatform's SCP81 [25], which specifies the adoption of the TSL for the NFC based mobile service architecture (i.e. TSM Framework discussed in section 2). Whereas, public key based Kerberos is suitable for the Multos application management architecture [35]. Table 4 show that the proposed protocol perform better than other listed protocols, which are either already adopted in case of the SCP81 or can be adopted in the smart card industry.

6 Conclusion and Future Research Directions

In this paper, we proposed a protocol referred as ACAP that provides the entity authentication, trust validation, mutual key and the contractual-agreement generation. The ACAP was then compared with existing protocols ranging from the internet-based protocols to ones that were specifically designed for the smart card environment. We have implemented the protocol and provided its performance measure. At the time of writing, authors were not aware of any other protocol that satisfies the same number of security and operational goals within the performance matrix of the ACAP.

As part of the future research direction, first we would like to provide a detailed formal analysis of the protocol. In addition, we consider that one of the important topics is how we can avoid the simulator problem and provide assurance to an SP that a smart card is a tamper-resistant, tamper-evident and a reliable device. In addition, we will look into the platform and runtime environment architecture that supports the TSM's and cardholder's space on the same device. Furthermore, we will analyse the prospects of extending the coopetitive framework to general purposes tamper-resistant devices.

References

1. Rankl, W., Effing, W.: Smart Card Handbook. John Wiley & Sons, Inc., NY (2003)
2. Girard, P.: Which Security Policy for Multiplication Smart Cards? In: Proceedings of the USENIX Workshop on Smartcard Technology, Berkeley, CA, USA, p. 3 (1999)
3. Akram, R.N., Markantonakis, K., Mayes, K.: A Paradigm Shift in Smart Card Ownership Model. In: Apduhan, B.O., Gervasi, O., Iglesias, A., Taniar, D., Gavrilova, M. (eds.): Proceedings of the 2010 International Conference on Computational Science and Its Applications (ICCSA 2010), pp. 191–200. IEEE CS, Fukuoka (2010)
4. Framework for Smart Card use in Government, Foundation for Information Policy Research, Consultation Response (1999)
5. Near Field Communication: The Keys to Truly Interoperable Communications, NFC Forum, White Paper (November 2006)
6. NFC Trials, Pilots, Tests and Live Services around the World. Online. NFC World
7. Pay-Buy-Mobile: Business Opportunity Analysis, GSM Association, White Paper 1.0 (November 2007)
8. Java Card Platform Specification, Sun Microsystem Inc. Std. Version 3.0.1 (May 2009)
9. Multos: The Multos Specification, Online
10. Trusted Module Specification 1.2, Trusted Computing Group Std., Rev. 103 (July 2007)
11. Akram, R.N., Markantonakis, K., Mayes, K.: A Dynamic and Ubiquitous Smart Card Security Assurance and Validation Mechanism. In: Rannenberg, K., Varadharajan, V., Weber, C. (eds.) SEC 2010. IFIP AICT, vol. 330, pp. 161–172. Springer, Heidelberg (2010)
12. Common Criteria for Information Technology Security Evaluation, Common Criteria Std. Version 3.1 (August 2006)
13. Sauveron, D., Dusart, P.: Which Trust Can Be Expected of the Common Criteria Certification at End-User Level? Future Generation Communication and Networking (2007)
14. Akram, R.N., Markantonakis, K., Mayes, K.: A privacy preserving application acquisition protocol. In: Geyong Min, F.G.M. (ed.) 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom 2012). IEEE Computer Society, Liverpool (2012)
15. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC (October 1996)
16. Akram, R.N., Markantonakis, K., Mayes, K.: Application Management Framework in User Centric Smart Card Ownership Model. In: Youm, H.Y., Yung, M. (eds.) WISA 2009. LNCS, vol. 5932, pp. 20–35. Springer, Heidelberg (2009)
17. Furlani, C.: FIPS 186-3 : Digital Signature Standard (DSS), Online, National Institute of Standards and Technology (NIST) Std. (June 2009)

18. Lepinski, M., Kent, S.: RFC 5114 - Additional Diffie-Hellman Groups for Use with IETF Standards (January 2008)
19. Diffie, W., Hellman, M.E.: New Directions in Cryptography. IEEE Transactions on Information Theory IT-22(6), 644–654 (1976)
20. GlobalPlatform: GlobalPlatform Card Specification, Version 2.2, GlobalPlatform Std. (March 2006)
21. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and Authenticated Key Exchanges. Des. Codes Cryptography 2, 107–125 (1992)
22. Aziz, A., Diffie, W.: Privacy And Authentication For Wireless Local Area Networks. IEEE Personal Communications 1, 25–31 (1994)
23. Horn, G., Martin, K.M., Mitchell, C.J.: Authentication Protocols for Mobile Network Environment Value-Added Services. IEEE Transactions on Vehicular Technology 51 (March 2002)
24. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A.D., Reingold, O.: Just Fast Keying: Key Agreement in a Hostile Internet. ACM Trans. Inf. Syst. Secur. 7 (May 2004)
25. Remote Application Management over HTTP, Card Specification v 2.2 - Amendment B, Online, GlobalPlatform Specification (September 2006)
26. Markantonakis, K., Mayes, K.: A Secure Channel Protocol for Multi-application Smart Cards based on Public Key Cryptography. In: Chadwick, D., Prennel, B. (eds.) Eight IFIP TC-6-11 Conference on Communications and Multimedia Security, pp. 79–96. Springer (September 2004)
27. Sirett, W.G., MacDonald, J.A., Mayes, K., Markantonakis, C.: Design, Installation and Execution of a Security Agent for Mobile Stations. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 1–15. Springer, Heidelberg (2006)
28. Dierks, T., Rescorla, E.: RFC 5246 - The Transport Layer Security (TLS) Protocol (August 2008)
29. Ryan, P., Schneider, S.: The Modelling and Analysis of Security Protocols: the CSP Approach. Addison-Wesley (2000)
30. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
31. FIPS 180-2: Secure Hash Standard (SHS), National Institute of Standards and Technology Std. (2002)
32. Urien, P.: Collaboration of SSL Smart Cards within the WEB2 Landscape. In: International Symposium on Collaborative Technologies and Systems, pp. 187–194 (2009)
33. Urien, P., Elrharbi, S.: Tandem Smart Cards: Enforcing Trust for TLS-Based Network Services. In: International Workshop on Applications and Services in Wireless Networks, pp. 96–104 (2008)
34. Harbitter, A., Menascé, D.A.: The Performance of Public Key-Enabled Kerberos Authentication in Mobile Computing Applications, pp. 78–85 (2001)
35. Multos: Guide to Loading and Deleting Applications, MAOSCO, Tech. Rep. (2006)