A Secure and Trusted Channel Protocol for the User Centric Smart Card Ownership Model

Abstract. The User Centric Smart Card Ownership Model (UCOM) provides an open and dynamic smart card environment enabling cardholders to request installation/deletion of an application to which they are entitled. As in this model, smart cards are not under the control of a centralised authority; hence, it is difficult for an application provider to ascertain their trustworthiness. At present, proposed secure channel protocols for the smart card environment do not provide adequate assurance required by the UCOM. In this paper, we explore the reasons behind their failure to meet the UCOM requirements and then propose a secure and trusted channel protocol that meets them. A comparison of the proposed protocol with existing smart card and selected Internet protocols is provided. Then we analyse the protocol with the CasperFDR tool. Finally, we detail the implementation and the performance measurement.

1 Introduction

The multi-application smart card initiative has been around since late 1990s; however, it did not experience a wide scale adoption [1]. Nevertheless, recently the introduction of the Near Field Communication (NFC) [2] technology and commercial realities have reinvigorated it [3].

This is an encouraging trend and there are a number of trials [4] being conducted around the world. In most of these trials, either the traditional ownership model termed as Issuer Centric Smart Card Ownership Model (ICOM) [5], or an extension of it referred to as Trusted Service Manager (TSM) [6] is deployed. In the ICOM, a centralised authority referred as card issuer has the sole control of the application installation and deletion on the issued smart cards.

The User Centric Smart Card Ownership Model (UCOM) [7] delegates the smart card's ownership to their respective users (cardholders). The term ownership means "*freedom of choice*" that only the users have the privilege to request installation or deletion of an application from a Service Provider (SP). Each SP has an Application Lease Policy (ALP) [8], which if satisfied by a smart card the SP would lease the respective application to it.

For leasing an application, the respective SP will establish a secure channel protocol with a smart card along with ascertaining its security and operational assurance. For this purpose, we propose a Secure and Trusted Channel Protocol (STCP) in this paper. The STCP not only establishes a secure communication channel but also provides assurance that the participating smart card is secure and trustworthy.

We start the discussion with a brief description of the related work and provide the rationale why we considered that most of the existing smart card protocols fall short in the UCOM environment. The discussion is then extended in section three to the smart card architecture in the UCOM environment that provides dynamic and ubiquitous security and reliability assurance. In section four, we propose the STCP followed by the informal and mechanical formal (CasperFDR) analysis of the STCP in section five. Section six details the performance measurements and finally, in section seven future research directions and conclusion of the paper are provided.

2 Secure Channel Protocols

In this section, we explore the rationale behind the STCP along with the related work that we use as a point of reference for later discussions.

2.1 Motivation

A Secure Communication Protocol (SCP) by definition provides either or both: entity authentication and key exchange between communicating parties (end points). The SCP preserves the

confidentiality and integrity of the messages on the channel but does not assure the same at the end points.

Nevertheless, there can be implicit assurance in the integrity and security of the end points as articulate by ETSI TS 102 412 [9]. This states that the smart card is a secure end point under the assumption that it is a tamper-resistant device. The implicit assumption is valid for the traditional smart card environment (e.g. ICOM and TSM) because smart cards are issued by "trusted" card issuers. The assurance is provided by the card issuer or in some cases indirectly by the Common Criteria (CC) evaluation [10]. This became the fundamental assumption in most of the smart card based SCPs. For the ICOM, this makes sense as the strict control of a smart card will effectively restrict it to only execute SCP with an entity that has prior authorisation from the card issuer.

On contrary, in the UCOM, there is no such authority (i.e. card issuer); hence, the assumption of the implicit assurance is no longer valid as illustrated by the simulator problem described in [11]. A simulated smart card on a computer can initiate a communication channel with an SP in the UCOM, download its application, and then try to reverse engineer it.

A trusted channel is a secure channel that is cryptographically bounded to the current state of the communicating parties [12]. This state can be a hardware and/or software configuration, and ideally it will require a trustworthy component to validate it to be the same as claimed. Such a component in most of the instances is a Trusted Platform Manager (TPM) [13] as demonstrated by [14] and [15].

In the UCOM, an SP does not have any prior trust relationship with a smart card. Therefore, the traditional smart card SCPs will fail to provide: (a) assurance that an SP is communicating with a genuine smart card platform and not a simulator, (b) the smart card security and operational environment is certified by a reputed third party evaluation, (c) the security and operational environment state is still valid as it was at the time of the evaluation, and (d) the smart card is owned by the user who is requesting the application download. These issues in the context of the UCOM are further discussed in section 2.3.

We define the STCP in the context of the UCOM as a protocol providing a secure and reliable communication channel coupled with an assurance of security and integrity concerning the communicating smart card. The STCP is used during: (a) application installation/deletion process, and/or (b) communication between an installed application and its respective SP.

2.2 Related Work

In this section, we restrict to the discussion of the protocols that are specifically proposed for the smart card environment or being used as point of comparison in later discussions.

Early smart card protocols were based on the symmetric key crypto-system like (deprecated) SCP01 of the GlobalPlatform specification [16]. Other protocols specified by the GlobalPlatform specification are SCP02 (based on Triple-DES), SCP10 (based on asymmetric key crypto-system) [16], SCP81 (based on SSL/TLS) [17], SCP03 (based on AES) [18], and SCP80 for the mobile telecom industry [19].

The concept of trusted channel protocols was put forward by Gasmi et al. [12] along with the adaptation of TLS protocol [20]. Later Armknecht et al. [15] proposed another adaptation of OpenSSL to accommodate the concept of the trusted channel; similarly, Zhou and Zhang [14] also proposed an SSL based trusted channel protocol.

In section 5.3, we will compare the proposed STCP with the existing protocols. These protocols include the Station-to-Station (STS) [21], the Aziz-Diffie (AD) protocol [22], the ASPeCT protocol [23], Just-Fast-Keying (JFK) [24], trusted TLS (T2LS) [12], GlobalPlatform SCP81 [17], Markantonakis-Mayes (MM) protocol [25], and Sirett-Mayes (SM) protocol [26]. The rationale for not including SCP10 is based on its similarity with the MM protocol.

The selection of the protocols is intentionally kept broad to include well-established protocols like STS, Aziz-Diffie (AD) and JFK protocols. Also including the ASPeCT protocol, which is

designed specifically for the mobile network's value-added services. The T2LS is based on the concept of trusted channels where SCP81, SM, and MM protocols are specific to smart cards.

2.3 Requirements and Goals of the STCP

The proposed protocol that supports the UCOM architecture should at minimum meet the following requirements and goals. The listed requirements are selected on the basis of the desirable features for an SCP and specific requirements of the UCOM.

G1) *Mutual Entity Authentication*: Both a smart card and an SP authenticates to each other to avoid masquerading by a malicious entity.
G2) Exchange of certified public keys between the entities to facilitate in the key generation and entity authentication process.
G3) Mutual Key Agreement: Communicating parties will agree on the generation of a key during the protocol run.
G4) Joint *Key Control*: Communicating parties will mutually control the generation of new keys to avoid one party choosing weak keys or predetermining any portion of the session key.
G5) *Key Freshness*: The generated key will be fresh to the protocol session to protect replay attacks.
G6) Mutual *Key Confirmation*: Communicating parties will provide implicit/explicit confirmation that they have generated the same keys during a protocol run.
G7) Known-Key Security: If a malicious user is able to obtain session key of a particular protocol run, it should not able him or her to retrieve the long-term secrets (*private keys,*) or *session keys* (future and past).
G8) Unknown *Key* Share Resilience: In the unknown key share attack, an entity $\mathcal{X}$ believes that it has shared a key with an $\mathcal{Y}$, where the entity $\mathcal{Y}$ mistakenly believes that it has shared the key with entity $\mathcal{Z} \neq \mathcal{X}$
G9) *Key* Compromise Impersonation (KCI) Resilience: If a malicious user retrieves the long-term key of an entity, it will enable him or her to impersonate the entity. Nevertheless, it should not facilitate him or her to impersonate other entities to it [27].
G10) *Perfect Forward Secrecy*: In case, if the long-term keys of communicating entities are compromised. It does not enable a malicious user to compromise previously generated session keys.
G11) Mutual *Non-Repudiation*: Communicating entities will not be able to deny that they have executed the protocol run with each other.
G12) Partial Chosen Key (PCK) Attack: Protocols that claim to provide joint key control are susceptible to this attack [28]. In this attack, if two entities provide separate values to the key generation function then one entity has to communicate its contribution value to the other. The second entity can then compute the value of its contribution in such a way that it can dictate its strength (able to generate a partially weak key). However, this attack depends upon the computational capabilities of the second entity.
G13) Trust Assurance (Trustworthiness): The communicating parties not only provide security and operation assurance but also validation proofs that are dynamically generated during the protocol execution [29].
G14) Memory-DoS and Computation-DoS Prevention: The protocol should not require the server (in our case SP's servers) to allocate the resources before authenticating and validating the state of the requesting entity (a smart card) or verifying the credentials of the authorised user.
G15) Privacy: A third party should not be able to know the identities of the user or her smart card, either over the Internet or Over-the-Air (OTA). In addition, during the trust validation and assurance process; the requesting SP should not be able to gain any additional information about the platform (e.g. applications installed on a smart card).

G16) Simulator Attack: This attack discussed in [29] allows a malicious user to masquerade a smart card platform on a computer (as a simulation). Such a possibility will enable the malicious user to download an application onto a simulated platform and then perform reverse engineering on the downloaded application: revealing proprietary and sensitive data of the application. Therefore, any proposed protocol for the UCOM architecture should integrate the platform attestation in its specification that verifies the current state of the platform to be trustworthy.

G17) Platform & Application User Separation (PAU) Attack: This attack is discussed in [29] in which a malicious user provides access credentials of a genuine user to an SP and downloads the respective application onto her smart card. Any protocol should tie a platform with its respective card-owner (user) to avoid platform & application user separation attack.

For formal definition of the terms (italicised) used in the above list, readers are advised to refer to [30]. The stated goals in this section are later used as point of reference to compare (see table 4) the proposed STCP with listed protocols in section 2.2.

3 Smart Card Assurance and Validation Mechanism

In this section, we describe the UCOM smart card architecture with on the emphasis on the trust assurance mechanism.

3.1 Smart Card Architecture

In the ICOM, both the card issuer and the application provider have an offline relationship. That may translate into having business and legally-binding contracts that define the terms and conditions for both parties to co-exist, and abide by each other's security requirements/policies on a smart card.

However, such an assumption is impossible to make in the UCOM where smart cards are not under any centralised authority. This absence of the prior and offline trust in the UCOM is the major cause why most of the existing protocols fail the UCOM requirements (section 2.3).An essential architectural change is required to the smart card architecture that can facilitate the establishment of the trustworthiness of both the smart card platform and (installed) application(s) that is referred to as Trusted Environment & Execution Manager (TEM) and illustrated in the overly simplified figure 1.

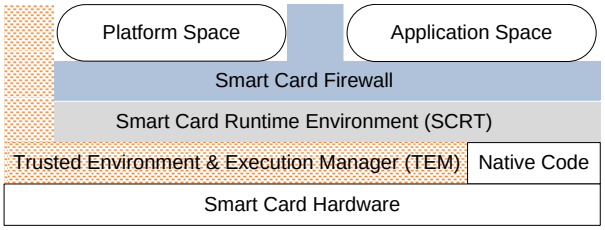

Fig. 1. Overview of the UCOM smart card architecture.

3.1.1 Trusted Environment & Execution Manager The TEM is used to provide a remote, and dynamic security assurance and validation that the smart card's state is as it was at the time of a third party evaluation. For security assurance, the third party evaluator issues an Product Evaluation Certificate (PEC), which is a cryptographically signed certificate that details the security and operational functionality of the evaluated product. The evaluation certificate will also certify a unique signature key pair of the card manufacturer. During the card manufacturing, the smart card will generate a signature key pair [30] that will be certified by the card manufacturer. The certificate hierarchy in the UCOM architecture is shown in figure 2. One thing to note is that at present CC [10] or any other evaluation scheme for that matter does not provide the PEC but proposals presented in [29] and [31] can be utilised.

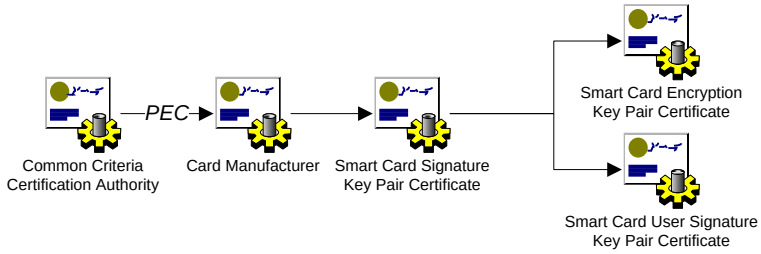

Fig. 2. Certificate hierarchy in the UCOM architecture.

For security validation, the TEM implements a validation mechanism that is divided into two parts: tamper-evidence and reliability assurance. Smart cards are required to be a tamper-resistant device [32] and for this purpose card manufacturers implement 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 on the other hand verify the software part of the smart card that is crucial for its security and reliability has not been tampered with.

A TEM tamper-evidence process should provide properties that are listed as below:

1. Robustness: On input of a certain data, it always produces the associated output.
2. Independence: When the same data is input to tamper-evidence process on two different devices, it outputs different values.
3. Pseudo-randomness: The generated output should be computationally difficult to distinguish from a pseudo-random function.
4. Tamper-evidence: An invasive attack to access the function should have irreversible changes, which render the device dead.
5. Assurance: The function can provide assurance (either implicitly or explicitly) to independent (non-related) verifiers. It should not require an active connection with the device manufacturer to provide the assurance. The assurance refers to the current hardware and software state is as it was at the time of third party evaluation.

Table 1. Comparison of different proposals for validation mechanism.

Features	**Active-Shield**	**KAT**	**Keyed-HMAC**	**PRNG**	**PUF**
Robustness	Yes	Yes	Yes	Yes	Yes
Independence	No	No	No	Yes	Yes
Pseudo-randomness	No	No	Yes	Yes	Yes
Tamper-evidence	Yes	No	–	-	Yes
Assurance	Yes	No	No	No	Yes*

Note: "Yes" means that the mechanism supports the feature. Similarly, "No" indicates that the mechanism does not support the required feature. The entry "Yes*" means that it may support the feature if adequately designed.

For the TEM tamper-evidence process there can be several candidates including using active (intelligent) shield/mesh [32], Known Answer Test (KAT) [33], hard-wired HMAC key, attestation based on PRNG [34] and Physically Unclonable Function (PUF) [35], etc. Based on the above listed features and table 1, we can safely assume that PUFs and PRNGs are better candidate for the validation mechanism. Therefore, we have included an algorithm in appendix B that implements PUF and PRNG based validation mechanism. We avoid discussing the details of the validation mechanism and its effectiveness in this paper as its beyond the scope of this paper.

4 Secure and Trusted Channel Protocol

In this section, we start the discussion with the notation used followed by the description of the proposed STCP.

4.1 Protocol Notation

The notation used in the protocol description is listed in table 2;

Table 2. Notation used in protocol description.

SP	:Denotes a Service Provider.
SC	:Denotes a smart card.
U	:Denotes a smart card owner (user).
X_i	:Represents the identity of an entity X.
g^{rX}	:Diffie-Hellman exponential generated by an entity X.
$Cert_X$	:Signature key certificate of an entity X.
SC_{Con}	:Capability list (i.e. supported security and operational functions) of the smart card.
X_{Sup}	:List of Diffie-Hellman groups and protocol parameters (e.g. cryptographic algorithms) supported by an entity X.
X_{Sel}	:List of Diffie-Hellman groups and protocol parameters selected by 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.
$[M]_{K_a}^{K_e}$	:Message M encrypted by the session encryption key K_e and then MAC is computed using the session MAC key K_a. Both keys K_e and K_a are generated during the protocol run.
$Sig_x(Z)$	:Signature generated on data Z by the entity x using a signature algorithm [36].
$H(Z)$	:Is the result of generating a hash of data Z.
$H_k(Z)$	:Result of generating a keyed hash of data Z using key k.
S_{Cookie}	:Session cookie generated by the respective SP. It indicates the session information and facilitates in protection against DoS attacks along with (possibly) providing the protocol session resumption facility.
U_{Cre}	:User authentication credential (e.g. login and password) associated with a particular SP.
VR	:Validation request send by an SP to a smart card. In response the smart card provides the security and reliabiltiy assurance to the SP.
ADP	:The ADP will include appropriate parameters for the application download protocol (which is beyond the scope of this paper).
ALP	:The ALP is the minimum security, reliability and operational requirements imposed by the respective SP on a smart card. The ALP also includes the respective application's details that include application size and functionality support requirements.

4.2 Pre-Protocol Process

To initiate the STCP protocol, a user will connect her smart card to internet through a device (e.g. mobile phone, desktop computer) that host Card Application Management Software (CAMS). The CAMS provides an interface between a smart card, user and an SP. The user will provide the CAMS with an SP's web address from where its applications can be downloaded. The CAMS connects the SP with the smart card and STCP is initiated.

4.3 The STCP Protocol

Protocol messages are listed in table 3 and described as below;

Table 3. Secure and Trusted Channel Protocol (STCP).

1. $SP \rightarrow SC : SP_i \| VR \| N_{SP} \| g^{rSP} \| SP_{Sup} \| S_{Cookie}$
2. $SC \rightarrow SP : g^{rSC} \| N_{SC} \| SC_{Sel} \| SC_{Con} \| [Sig_{SC}(SC_i \| SP_i \| g^{rSP} \| g^{rSC} \| N_{SP} \| N_{SC}) \| Cert_{SC}]_{K_a}^{K_e} \| S_{Cookie}$
3. $SP \rightarrow SC : [VR \| ADP \| Sig_{SP}(SP_i \| SC_i \| g^{rSP} \| g^{rSC} \| N_{SP} \| N_{SC} \| ALP) \| Cert_{SP}]_{K_a}^{K_e} \| S_{Cookie}$
4. $SC \rightarrow SP : [U_{Cre} \| Sig_U(SC_i \| SP_i \| U_i \| S_{Cookie}) \| Cert_U]_{K_a}^{K_e} \| S_{Cookie}$

Message 1: The SP generates a random number N_{SP} and computes the Diffie-Hellman exponential g^{rSP}. The SP_{Sup} will also include the user's authentication process details in addition to the ones described in table 2. These details communicate to the smart card the way the SP would like to perform the user authentication. The MAC "$H_{SP_k}(g^{rSP} \| N_{SP} \| SC_{IP})$" serves as a session cookie "S_{Cookie}", and it is appended with each subsequent message sent by the smart

card. It indicates the session information and facilitates the protection against DoS attacks along with (possibly) providing the protocol session resumption facility. Finally, the SP will request the smart to provide assurance that its current state is same as it was at the time of third party evaluation by sending the VR.

If the respective smart card does not support the Diffie-Hellman group selected by the SP, then it will send a rejection message including a list of groups supported by the smart card "SC_{Sup}". If the smart card support the selected group, then it will proceed with the second message.

Message 2: The smart card in response generates a random number, and Diffie-Hellman exponential g^{rSC}. It can then calculate the $k_{DH} = (g^{rSP})^{rSC}$ $(mod\ n)$ which will be the shared secret from which the rest of keys will be generated. The encryption key is generated as $K_e = H_{k_{DH}}(N_{SP}\|N_{SC}\|''1'')$ and MAC key as $K_a = H_{k_{DH}}(N_{SP}\|N_{SC}\|''2'')$. We can further generate (session) keys in a similar manner for the application download protocol or for the application that requested it.

Subsequently, the TEM executes the validation mechanism, which if successful will generate signature over the identities of the smart card and SP, along with the smart card random numbers and Diffie-Hellman exponentials. This message signed by the smart card signature key is referred as the validation proof. As the signature key will be different if state of the platform is modified; therefore, by verifying the signature the SP can ascertain the current state of the platform.

On receipt of this message, the SP will first generate the session keys then verify the smart card certificate. The smart card certificate gives the required assurance that the smart card is evaluated by a third party that meets the SP's ALP requirements, followed by the verification of the signature. The assumption here is, if the third party evaluation has concluded as the smart card is a tamper-resistant device that implements an effective validation mechanism. Then, this will mean the TEM keys are difficult to be revealed to a malicious user. Hence, in the presence of tamper-evidence mechanism only a genuine TEM can generate the correct signature.

As a next step, the SP ascertains whether it has already issued an application lease to the stated smart card. If there is an application lease to the smart card, the protocol will terminate otherwise the SP will proceed to the next step. Furthermore, the SP will check the values of Diffie-Hellman exponentials (i.e. g^{rSP} and g^{rSC}) and generated random numbers to avoid main-in-the-middle and replay attacks.

Message 3: The SP will then sign the random numbers and identities of both the SC and SP along with the ALP.

On receipt, the smart card will verify whether it can support the listed requirements in the ALP. Furthermore, the smart card will also check the Diffie-Hellman exponentials and generated random numbers to avoid main-in-the-middle and replay attacks.

Message 4: The smart card requests the cardholder to provide the SP's authentication credentials as requested by the SP in the SP_{Sup}. After the user provides those credentials they are packaged as U_{Cre} that is concatenated by a signed message containing identities of the smart card (unique serial number [32]), SP and user along with the session cookie. Reason behind including the session cookie in the signature is to provide a proof signed by the user's signature key pair that she initiated the session represented by the session cookie details. The signed message along with the certificate and user's credentials are then encrypted and MACed.

The SP verifies the U_{Cre} and if the user authenticates then it will verify whether the user (owner) identity refereed in the $Cert_U$ is the one of an authorised and authenticated user. If so, then the SP will verify the signature. Furthermore, the U_{Cre} will provide the SP an assurance that the user is cryptographically bounded with the smart card (i.e. has the ownership of the smart card).

4.4 Post-Protocol Process

On its successful completion, not only the STCP will provide an SP the assurance that the requesting smart card is suitable for its application but also generates the key material for the application download process. For completeness, an SP and a smart card can use a symmetric key application download protocol from the GlobalPlatform specification [16].

5 Protocol Analysis

In this section, we discuss the proposed STCP in terms of informal, and formal mechanical analysis along with detailing the performance results.

5.1 Brief Informal Analysis of the Protocol

In this section, we informally discuss requirements for the STCP grouping first twelve together and treating the rest individually.

5.1.1 Requirements One to Twelve. In this section, we constantly refer to the protocol requirements and goals in section 2.3 with their respective numbers as listed in the same section. Therefore, here onward in this section any reference to a goal or requirement number refers to the listed item in section 2.3.

During the STCP protocol, in message two and three the communicating entities authenticate to each other. The respective user is authenticated to the SP in message four; thus, satisfying the G1. Similarly, for G2, all communicating entities exchange cryptographic certificates to facilitate in authentication and ownership proof (in case of user signature key certificate).

The proposed STCP satisfies the requirement G3, G4, G5 and G12 by first requiring the SP to generate the Diffie-Hellman exponentials as it is computationally powerful than the smart card. If the smart card generate the exponential first then the SP can choose a weak key. Whereas, , as smart cards are computationally restricted device they cannot perform such a task.

In the STCP, session keys generated in one session has no link with the session keys generated in other session, even when the session is established between the same entities. This enables the protocol to provide resilience against the known-key security (G7). This unlinkability of session key is based on the fact that each entity not only generate a new Diffie-Hellman exponential but also a random number which are used during the STCP. Therefore, even if an adversary "$\mathcal{A}$" finds out about the exponential and random numbers of a particular session; it would not enable him to generate past or future session keys.

Furthermore, to provide unknown key share resilience (G8) that STCP includes the Diffie-Hellman exponentials along with generated random numbers and each communicating entity then signs them. Therefore, the receiving entity can then ascertain the identity of the entity with whom it has shared the key.

The STCP can be considered to be a KCI resilient (G9) protocol, as protection against the KCI is based on the digital signatures. In addition, the cryptographic certificates of each signature key also include its association with a particular SP or a smart card. Therefore, if $\mathcal{A}$ has the knowledge of the signature key of a smart card (or an SP) then it can only masquerade the smart card to other entities but not other entities to the smart card.

The STCP also meet the perfect forward secrecy (G10) by making the key generation process independent of any long term keys. The session keys are generated using fresh values of Diffie-Hellman exponentials and random numbers, regardless of the long term keys like smart card, user and SP's signature keys. Therefore, even if $\mathcal{A}$ finds out the signature key of any entity it will not enable him to find out past session keys. This independence of long term secrets from the session key generation process also enabled the STCP to satisfy the G7.

Communicating entities in the STCP share signed messages with each other that includes the session information; thus, providing mutual non-repudiation (G11).

5.1.2 Trust Assurance (Trustworthiness). In the proposed STCP, only smart cards provide the assurance of their current state to be secure and trustworthy to respective SPs; not the other way around. The reason behind this is the deployment environment of the STCP where smart cards are inherently not trustworthy and the UCOM not requiring the trustworthiness of an SP. The UCOM assumes that an SP can be malicious but it will translate into the lease of a malicious application(s). Therefore, security and reliability analysis (e.g. bytecode verification [37]) of the downloaded application and not the respective SP is required. Details of which are omitted in this paper as they fall beyond the scope of the STCP.

A trusted channel establishment between a smart card and an SP is based on the security and trustworthiness of the TEM. The argument for the trust goes as; the respective smart card manufacturer gets a particular batch of smart cards certified from a third party evaluator. As discussed in section 3.1.1, the evaluation facility will issue a certificate for the evaluated product to the respective manufacturer. That in return will mass-produce the smart cards that comply with the evaluation certificate. The TEM security validation mechanism is also evaluated, and during the STCP, the SP will validate the hardware and software (e.g Smart Card Operating System: SCOS).

If and only if the validation mechanism is successful that the TEM will generate the signature on the test results. On receipt of these results; an SP can also verify the test results and validate the certificate chain (i.e. to check the evaluation authority of the smart card). In case it trusts the evaluation authority and current state validates that the smart card complies with the evaluation, then it will continue the protocol, otherwise it is terminated.

Therefore, the trust in the established protocol session comes from assurance that the smart card is still in compliance with the evaluated state. That is certified to be secure and trustworthy by a third party evaluation. In which the respective SP has implicit or explicit trust.

5.1.3 Denial-of-Service Protection. The aim of the DoS protection is to provide a level of assurance that the proposed protocol might not be used to mount a DoS attack against the SP. This is achieved by (a) adding a session cookie to the protocol messages that serves as the session identifier (e.g. $H_{SP_k}(g^{r_{SP}} \| N_{SP} \| SC_{IP})$), which includes the smart card's IP address, and (b) by not requiring the SP to perform any public key operations unless it receives user or platform authentication.

The session cookie is generated by the respective SP and it is smart card's responsibility to include the cookie in every message that it sends to the SP. On receiving a message from a smart card, the SP verifies the session cookie and if it belongs to an active session, then it can ascertain that the message came from a genuine host and not from an entity that is trying to mount the DoS attack.

The second feature, which does not require the SP to perform any heavy computations until the smart card does not provide a signed message either by the user's or platform's signature key. This is necessary to avoid the SP to commit memory and computational resources; unless the communicating smart card is authenticated to the SP's.

5.1.4 Privacy. The privacy preservation goal of the STCP requires that the privacy of the user should be protected. This requirement does not include the privacy for the SP as part of their business model is to advertise their presence and identity (i.e. web servers). Therefore, the privacy requirement is restricted to the preservation of the user's and her smart cards identity. The smart card's identity is protected to avoid traceability. With traceability, we mean that if a user acquires an application from a malicious SP then it knows the smart card's identity. In the future, if the user tries to acquire an application from another SP using the same smart card, the malicious SP eavesdropping on the communication channel might trace it back to the user. In the proposed protocol, we do not send any information that can be uniquely attached to a

particular user or a smart card in plaintext. All communications that include the identities and cryptographic certificates are encrypted.

However, if a user always gets online through a permanent connection (i.e. fixed Internet Protocol address) then a malicious user can trace the communication to a user. Only if the malicious user has previously recorded the association of the IP address with the respective user. In such a scenario, privacy preservation is difficult to maintain in a restricted framework of the secure channel protocols; therefore, the proposed STCP does not provide the protection against traceability under fixed uniquely associated IP addresses to users.

5.1.5 Simulator Protection. The proposed STCP provides protection in relation to the simulator attack: relying on the TEM's operations, trustworthiness, and effectiveness of the evaluation laboratory. The certification assures that the smart card is tamper-resistant, and it is highly unlikely that a malicious user can retrieve the signature key pair of the smart card. In addition, it also assures that the TEM validation mechanism is effective and reliable.

This will in theory give the assurance to the respective SP that the smart card with whom it is communicating is not a simulator, and the current state of the smart card is as it was at the time of evaluation. It does not mean that it can still be secure or a malicious user is not able to simulate the environment with a genuine TEM signature key pair. It only gives the assurance that the smart card is secure against the attacks it was evaluated by the third party as stated in the issued certificate [29] and is state-of-the-art tamper-resistant device at the time of evaluation. Therefore, if the evaluation certificate does not meet SPs requirements or it out-dates the current attacker capability then the respective SP should decline the application lease. As stated earlier, granting an application lease is on the sole discretion of the respective SP, so if they are not satisfied with the requesting smart card, they should not lease the application.

5.1.6 Platform & Application User Separation Attack. As discussed in section 2.3, in this attack a malicious user tries to install an application that belongs to some other user on her smart card. Therefore, the identity of the card owner and the leaseholder of the application are different.

The STCP provides assurances against this attack as in the second message the card owner's identity and ownership proof is included in the message. The ownership proof comes from the signature generated using the smart card owner's signature key pair (figure 2). Thus, providing a cryptographic binding between the smart card and its current owner.

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 [38] 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) [39]. The CSP description of the protocol can be machine-verified using the Failures-Divergence Refinement (FDR) model checker [40]. The intruder's capability modelled in the Casper script (appendix A) for the proposed protocol is: 1) an intruder can masquerade any entity in the network, 2) intruder can read the messages transmitted in the network, and 3) an intruder cannot influence the internal process of an entity in the network.

The security specification for which the CasperFDR evaluates the network is as shown below. The listed specifications are defined in the #Specification section of appendix A: 1) the protocol run is fresh and both applications were alive, 2) the key generated by the server application is known only to the client application, 3) entities mutually authenticate each other and have mutual key assurance at the conclusion of the protocol, 4) long terms keys of communicating entities are not compromised, and 5) intruder is unable to deduce the identities of the user or the

smart card from observing the protocol messages. The CasperFDR tool evaluated the protocol and did not find any feasible attack(s).

5.3 Revisiting the Requirements and Goals

Table 4 provides the comparison between the listed protocols in section 2.2 with the proposed protocol in terms of the required goals (see section 2.3).

Table 4. Protocol comparison on the basis of the stated goals (see section 2.3.)

Goals	Protocols								
	STS	AD	ASPeCT	JFK	T2LS	SCP81	MM	SM	STCP
1. Mutual Entity Authentication	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$-*$	$*$
2. Exchange Certificates	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
3. Mutual Key Agreement	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
4. Joint Key Control	$*$	$*$	$*$	$*$	$*$	$*$			$*$
5. Key Freshness	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
6. Mutual Key Confirmation	$*$		$*$	$*$			$*$	$-*$	$*$
7. Known-Key Security	$*$	$*$	$*$	$*$	$*$	$*$	$*$		$*$
8. Unknown Key Share Resilience	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$-*$	$*$
9. KCI Resilience	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
10. Perfect Forward Secrecy	$*$		$*$	$*$	$*$	$*$			$*$
11. Mutual Non-Repudiation	$*$	$(*)$	$+*$	$*$	$*$	$*$	$+*$	$+*$	$*$
12. PCK Attack Resilience	$(*)$	$(*)$		$(*)$	$(*)$	$(*)$			$*$
13. Trust Assurance					$*$	$-*$			$*$
14. DoS Prevention				$*$					$*$
15. Privacy	$(*)$		$*$	$*$					$*$
16. Simulator Attack Resilience					$-*$				$*$
17. PAU Attack Resilience									$*$

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.

As shown in the table 4, the STS protocol meets the first eleven goals. The main issue with the STS protocol is that it does not provide adequate protection against partial chosen key attack (G12). The remaining goals are not met by the STS because of the design architecture and deployment environment, which did not require these goals. Similarly, the AD protocol does not meet G6, and G10-G17. In the AD protocol, the user reveals her identity by sending the user certificate in clear along with non-existence of key confirmation.

The most promising results were from the ASPeCT and JFK protocols that meet a large set of goals. Both of these protocols can be easily modified to provide the trust assurance (requiring additional signature). Furthermore, both of these protocols are vulnerable to the partial chosen key attacks. However, in the table 4 we opt for the possibility that the JFK can be modified to meet this goal. The reason behind this is based on the entity that takes the initiators role and if in the JFK we opt for the assumption that an SP will always take the initiators role then this goal is met by the JKF.

The T2LS protocol meets the trust assurance goal by default but because it is based on the TLS protocol, which does not meet most of the requirements of the STCP, so the T2LS also does not meet them. 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 card issuer; thus, implicitly trusting the respective smart card. Most of these protocols to some extent have the similar architecture in which a server generates the key and then communicates that key to the client (i.e. read smart card). There is no non-repudiation as they do not use signatures in the protocol run.

As apparent from the table 4, the proposed STCP satisfies all goals that were described in section 2.3. The protocols that are proposed specifically for the smart card environment only meet half of the stated goals. However, the security requirements for the UCOM are more stringent

Table 5. Protocol performance measures (milliseconds).

Measures	SSL	TLS	Kerberos	STCP	
				Card One	Card Two
Card Specification	32bit	32bit	32bit	16bit	16bit
Average	4200	4300	4240	3395.17	3532.09
Best Run	NA	NA	NA	3343	3359
Worse Run	NA	NA	NA	3875	6797
Standard Deviation	NA	NA	NA	69.04	134.28

Note: Above mentioned measurement values for SSL are taken from [46], TLS from [47] and Kerberos from [44].

than the ICOM. Nevertheless, we still consider that the proposed STCP should be deployed even in the ICOM and especially with any future ownership model that will support multi-applications on a smart card under the Trusted Service Manager (TSM) architecture.

6 Practical Implementation

The proposed protocol in section 4 do not specify actual details of the cryptographic algorithms, which are left to the respective SPs and smart cards. However, in our implementation we use AES (128-bit key) in Cipher Block Chaining (CBC) mode without padding [41] for both encryption and generating MACs. The signature algorithm was chosen to be RSA with 512-bit key. For generating hash values, we use SHA-256 [42]. For Diffie-Hellman key generation, we chose the 2058-bit MODP group specified in the RFC-5114 [43].

Our implementation model has two entities; a smart card, and an SP, implemented on a Java Card and 1.83GHz with 2GB RAM laptop, respectively. We have implemented two applets on 16bit Java Cards; one takes the role of the TEM and second as the protocol handler. At the time of writing the paper, we did not have access to the smart card platform that will enable us to implement the TEM close to the hardware level (see figure 1). We suffice our implementation at the application level and consider that similar or better performance can be attained if TEM is implemented as part of the platform (which we plan to do in future). At the application level, implementation of the TEM cannot have memory access to measure the hash value of the SCOS. Therefore, we generated the hash values on a fixed set of values stored in an array of size 256 bytes to represent an SCOS.

The performance of the raw implementation without any pre-computation, measured on two different Java Cards is listed in the table 5. The implementation of the protocol on Java Cards takes 9799 bytes. The values in the table 5 were taken from the data set that was collected by executing each protocol on individual cards for 1000 times and recording the time it takes to complete each iteration. We choose performance measures of the SSL, TLS, and public key based Kerberos [44] to provide a comparison with our proposal. The rationale for this choice is based on the GlobalPlatform's SCP81 for the TSM architecture (based on the SSL/TLS) and Multos application installation architecture [45] that can adopt the public key based Kerberos.

7 Conclusion and Future Research Directions

In this paper, we have proposed a protocol that provides a secure and trusted communication channel to the communicating parties. The proposed protocol was compared with nine other protocols against the stated goals and requirements, and it performed better then the selected protocols. We provide the mechanical formal analysis of the STCP that did not find any feasible attacks, and then finally we showed that it performs better then other protocols that are proposed for the TSM architecture. We consider that the proposed protocol is not only suitable for the UCOM architecture but we recommend it for the ICOM or TSM architectures.

As part of future research, the TEM architecture has to be formalised by defining how a PUF or another mechanism can provide assurance of hardware protection. Furthermore, we will look into how the UCOM architecture can be expanded into the other computing domains like mobile phones, tablets, and personal computers through a portable, cross-platform, fault-tolerant, and tamper-resistant generic device that is in a user's control.

References

1. D. Deville, A. Galland, G. Grimaud, and S. Jean, "Smart Card Operating Systems: Past, Present and Future," in *In Proceedings of the 5 th NORDU/USENIX Conference*, 2003.
2. *ISO/IEC 18092: Near Field Communication - Interface and Protocol (NFCIP-1)*, International Organization for Standardization (ISO) Std., April 2004.
3. "Co-Branded Multi-Application Contactless Cards for Transit and Financial Payment," Smart Card Alliance, USA, White Paper TC-08001, March 2008.
4. (2011) NFC Trials, Pilots, Tests and Live Services around the World. Online. NFC World.
5. D. Sauveron, "Multiapplication Smart Card: Towards an Open Smart Card?" *Inf. Secur. Tech. Rep.*, vol. 14, no. 2, pp. 70–78, 2009.
6. "Mobile NFC Services," GSM Association, White Paper Version 1.0, 2007.
7. R. N. Akram, K. Markantonakis, and K. Mayes, "A Paradigm Shift in Smart Card Ownership Model," in *Proceedings of the 2010 International Conference on Computational Science and Its Applications (ICCSA 2010)*, B. O. Apduhan, O. Gervasi, A. Iglesias, D. Taniar, and M. Gavrilova, Eds. Fukuoka, Japan: IEEE CS, March 2010, pp. 191–200.
8. ——, "Application Management Framework in User Centric Smart Card Ownership Model," in *The 10th International Workshop on Information Security Applications (WISA09)*, ser. LNCS, H. Y. YOUM and M. Yung, Eds., vol. 5932/2009. Busan, Korea: Springer, August 2009, pp. 20–35.
9. "Smart Cards; Smart Card Platform Requirements Stage 1(Release 9)," ETSI, France, Tech. Rep. ETSI TS 102 412 (V9.1.0), June 2009.
10. *Common Criteria for Information Technology Security Evaluation*, Online, Common Criteria Specification Version 3.1, August 2006.
11. R. N. Akram, K. Markantonakis, and K. Mayes, "Simulator Problem in User Centric Smart Card Ownership Model," in *6th IEEE/IFIP International Symposium on Trusted Computing and Communications (TrustCom-10)*, H. Y. Tang and X. Fu, Eds. HongKong, China: IEEE CS, December 2010.
12. Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, and N. Asokan, "Beyond Secure Channels," in *STC '07: Proceedings of the 2007 ACM workshop on Scalable trusted computing*. New York, NY, USA: ACM, 2007, pp. 30–40.
13. *Trusted Module Specification 1.2*, Trusted Computing Group Std., Rev. 103, July 2007.
14. L. Zhou and Z. Zhang, "Trusted Channels with Password-Based Authentication and TPM-Based Attestation," *International Conference on Communications and Mobile Computing*, pp. 223–227, 2010.
15. F. Armknecht, Y. Gasmi, A.-R. Sadeghi, P. Stewin, M. Unger, G. Ramunno, and D. Vernizzi, "An efficient implementation of trusted channels based on openssl," in *Proceedings of the 3rd ACM workshop on Scalable trusted computing*, ser. STC '08. New York, NY, USA: ACM, 2008, pp. 41–50.
16. *GlobalPlatform: GlobalPlatform Card Specification, Version 2.2,*, Online, GlobalPlatform Specification, March 2006.
17. *Remote Application Management over HTTP*, Online, GlobalPlatform Specification, September 2006.
18. *GlobalPlatform Card Technology: Secure Channel Protocol 03*, Online, GlobalPlatform Public Release, September 2009.
19. "Smart Cards; Secured Packet Structure for UICC based Applications (Release 6)," ETSI, France, Tech. Rep. ETSI TS 102 225 (V6.8.0), April 2006.
20. T. Dierks and E. Rescorla, "RFC 5246 - The Transport Layer Security (TLS) Protocol v1.2," Tech. Rep., August 2008.
21. W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and Authenticated Key Exchanges," *Des. Codes Cryptography*, vol. 2, pp. 107–125, June 1992.
22. A. Aziz and W. Diffie, "Privacy And Authentication For Wireless Local Area Networks," *IEEE Personal Communications*, vol. 1, pp. 25–31, First Quarter 1994.
23. G. Horn and B. Preneel, "Authentication and payment in future mobile systems," in *Computer Security Ů ESORICS 98*, ser. LNCS, J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, Eds. Springer, 1998, vol. 1485, pp. 277–293, 10.1007/BFb0055870.
24. W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold, "Just fast keying: Key agreement in a hostile internet," *ACM Trans. Inf. Syst. Secur.*, vol. 7, pp. 242–273, May 2004.
25. K. Markantonakis and K. Mayes, "A Secure Channel Protocol for Multi-application Smart Cards based on Public Key Cryptography," in *CMS 2004 - Eight IFIP TC-6-11 Conference on Communications and Multimedia Security*, D. Chadwick and B. Prennel, Eds. Springer, Sep 2004, pp. 79–96.
26. W. G. Sirett, J. A. MacDonald, K. Mayes, and C. Markantonakis, "Design, Installation and Execution of a Security Agent for Mobile Stations," in *Smart Card Research and Advanced Applications, 7th IFIP WG 8.8/11.2 International Conference, CARDIS*, ser. LNCS, J. Domingo-Ferrer, J. Posegga, and D. Schreckling, Eds., vol. 3928. Tarragona, Spain: Springer, April 2006, pp. 1–15.
27. S. Blake-Wilson, D. Johnson, and A. Menezes, "Key Agreement Protocols and Their Security Analysis," in *Proceedings of the 6th IMA International Conference on Cryptography and Coding*. London, UK: Springer, 1997, pp. 30–45.

28. C. Mitchell, M. Ward, and P. Wilson, "Key control in key agreement protocols," *Electronics Letters*, vol. 34, no. 10, pp. 980 –981, May 1998.
29. Anonymous.
30. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC, October 1996.
31. D. Sauveron and P. Dusart, "Which Trust Can Be Expected of the Common Criteria Certification at End-User Level?" *Future Generation Communication and Networking*, 2007.
32. W. Rankl and W. Effing, *Smart Card Handbook*. New York, NY, USA: John Wiley & Sons, Inc., 2003.
33. *FIPS 140-2: Security Requirements for Cryptographic Modules*, Online, National Institute of Standards and Technology (NIST) Std., Rev. Supercedes FIPS PUB 140-1, May 2005.
34. R. Kennell and L. H. Jamieson, "Establishing the genuinity of remote computer systems," in *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*. Berkeley, CA, USA: USENIX Association, 2003, pp. 21–21.
35. B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM conference on Computer and communications security*, ser. CCS '02. New York, NY, USA: ACM, 2002, pp. 148–160.
36. *FIPS 186-3 : Digital Signature Standard (DSS)*, Online, National Institute of Standards and Technology (NIST) Std., June 2009.
37. D. A. Basin, S. Friedrich, J. Posegga, and H. Vogt, "Java Bytecode Verification by Model Checking," in *CAV '99: Proceedings of the 11th International Conference on Computer Aided Verification*. London, UK: Springer, 1999, pp. 491–494.
38. G. Lowe, "Casper: a compiler for the analysis of security protocols," *J. Comput. Secur.*, vol. 6, pp. 53–84, January 1998.
39. C. A. R. Hoare, *Communicating sequential processes*. New York, NY, USA: ACM, 1978, vol. 21, no. 8.
40. P. Ryan and S. Schneider, *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley Professional, 2000.
41. Joan Daemen and Vincent Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Berlin, Heidelberg, New York: Springer Verlag, 2002.
42. *FIPS 180-2: Secure Hash Standard (SHS)*, National Institute of Standards and Technology (NIST) Std., 2002.
43. M. Lepinski and S. Kent, "RFC 5114 - Additional Diffie-Hellman Groups for Use with IETF Standards," Tech. Rep., January 2008.
44. A. Harbitter and D. A. Menascé, "The performance of public key-enabled kerberos authentication in mobile computing applications," pp. 78–85, 2001.
45. "Multos: The Multos Specification„" Online.
46. P. Urien, "Collaboration of SSL Smart Cards within the WEB2 Landscape," *Collaborative Technologies and Systems, International Symposium on*, vol. 0, pp. 187–194, 2009.
47. P. Urien and S. Elrharbi, "Tandem Smart Cards: Enforcing Trust for TLS-Based Network Services," *Applications and Services in Wireless Networks, International Workshop on*, vol. 0, pp. 96–104, 2008.

A CasperFDR Script

```
#Free variables
datatype Field = Gen | Exp(Field, Num) unwinding 2
halfkeySP, halfkeySC, iMsg, rMsg, EnMaKey : Field
SC, SP, User: Agent
gSC, gSP: Num
nSC, nSP, SCOSHash: Nonce
VKey: Agent->PublicKey
SKey: Agent->SecretKey
InverseKeys = (VKey, SKey), (EnMaKey, EnMaKey), (Gen, Gen), (Exp, Exp)

#Protocol description
0.    -> SP : SC
[SC!=SP]
<iMsg := Exp(Gen,gSP)>
1. SP -> SC : SP, nSP, iMsg%halfkeySP
<EnMaKey := Exp(halfkeySP, gSC); rMsg := Exp(Gen,gSC)>
2. SC -> SP : nSC, rMsg%halfkeySC
<EnMaKey := Exp(halfkeySC, gSP)>
```

```
3. SP -> SC: nSP, nSC
4. SC -> SP : {{rMsg, User, nSP}{SKey(User)}}{EnMaKey}
[rMsg==halfkeySP]
5. SP -> SC : {{iMsg,SP, nSC}{SKey(SP)}}{EnMaKey}
[iMsg==halfkeySC]
6.SC -> SP : {{SCOSHash, SC, nSP}{SKey(SC)}}{EnMaKey}

#Actual variables
SCard, SProvider, USER, MaliciousEntity: Agent
GSC, GSP, GMalicious: Num
NSC, NSP, SCOSHASH, NMalicious: Nonce

#Processes
INITIATOR(SP,SC, User, gSP, nSP)knows SKey(SP), VKey
RESPONDER(SC,SP, User, SCOSHash, gSC, nSC) knows SKey(User), SKey(SC), VKey

#System
INITIATOR(SProvider, SCard, USER, GSP, NSP)
RESPONDER(SCard, SProvider, USER, SCOSHASH, GSC, NSC)

#Functions
symbolic VKey, SKey

#Intruder Information
Intruder = MaliciousEntity
IntruderKnowledge = {SProvider, SCard, MaliciousEntity,
GMalicious, NMalicious, SKey(MaliciousEntity), VKey}

#Specification
Aliveness(SP, SC)
Aliveness(SC, SP)
Agreement(SP, SC, [EnMaKey])
Secret(SP, EnMaKey, [SC])
Secret(SC, EnMaKey, [SP])
Secret(SC, User, [SP])

#Equivalences
forall x, y : Num . Exp(Exp(Gen, x), y) = Exp(Exp(Gen, y), x)
```

B Attestation Process

Algorithm 1: Attestation process based on a PUF and a PRNG

Input :
l: list of selected memory addresses.
hK: hard-wired key that would be destroyed in an invasive attack.

Output:
S: signature key of the $\mathcal{SC}$.

Data:
seed: temporary input value for the PUF set to zero.
n: number of memory addresses in the list l.
i: counter set to zero.
a: memory address.
$prKey$; PRNG secret key that is unique to each smart card.
k: secret key used to encrypt the smart card signature key.
S_e: encrypted signature key S using a symmetric algorithm with key k.

```
AttestationHandler (l, hK) begin
    while i < n do
        a ⟵ Read (i, l)
        seed ⟵ GenHash (ReadMemoryContents (a), hK,seed)
        i ⟵ i+1
    if seed ≠ ∅ then
        if Attestation == PUF then
            k ⟵ PUF (seed)
        if Attestation == PRNG then
            k ⟵ PRNG (seed, prKey)
    else
        return testfailed
    S ⟵ DecryptionFunction (k, S_e)
    return S
```