Simulator Problem in User Centric Smart Card Ownership Model

Raja Naeem Akram, Konstantinos Markantonakis, and Keith Mayes
Information Security Group, Smart Card Centre, Royal Holloway, University of London.
Egham, United Kingdom. Email: {R.N.Akram, K.Markantonakis, Keith.Mayes}@rhul.ac.uk

Abstract—The Issuer Centric Smart Card Ownership Model (ICOM) gives complete control of smart cards to their respective card issuers, enabling them to install, modify or delete applications remotely, in a secure manner. However, the User Centric Smart Card Ownership Model (UCOM) delegates the ownership of smart cards to their users, entitling them to install or delete any application according to their requirements. In the UCOM there might be no off-card relationship between a smart card and an application provider, referred to as a Service Provider, which is the cornerstone of the UCOM security framework. Therefore, this creates unique security issues like the simulator problem, in which a malicious user may simulate the smart card environment on a computing device and requests installation of an application. Following this, it might be possible to retrieve sensitive application data by reverse engineering. In this paper, we analyse the simulator problem, how it affects the UCOM and propose a possible solution.

I. INTRODUCTION

A multi-application smart card initiative enables interrelated and corroborative applications from diverse industries [1] that co-exist and augment each other’s functionality. Until recently, it was not widely deployed because of issues relating to ownership, branding, and business objectives. However, Near Field Communication (NFC) [2] has invigorated the convergence of different services (applications) onto a single smart card.

In most of the NFC-based field trials, the adopted ownership model could be considered an extension of the Issuer Centric Smart Card Ownership Model (ICOM), and is based on the concept of the “Trusted Service Manager” (TSM) [3]. TSMs have administrative authority on smart cards, and no application can be installed or deleted without their prior authorisation [4]–[6]. The main difference from the traditional ICOM is that the TSMs might be a neutral card management authority and would not install their application onto the cards [7]. This trend is encouraging, as it is attracting commercial interest. However, we consider that a true multi-application smart card would give the freedom to choose an application to cardholders without restricting it to a centralised authority.

Contrary to the ICOM, the User Centric Smart Card Ownership Model (UCOM) delegates the ownership of smart cards to their users. The term “ownership” means the freedom of “choice” of applications that users can install or delete on their smart cards, in a ubiquitous and seamless manner [8]. New frameworks create unique security issues of their own though, and the UCOM is no exception. Allowing users to install applications as they desire raises distinctive security issues, one of which is the simulator problem, which is the focus of this paper.

Section two briefly discusses the ICOM and UCOM. In section three, the simulator problem, possible attack scenarios and the requirements that a solution should satisfy are described. A proposed solution to the simulator problem is detailed in section four. We analyse the proposal in section five. Section six lists future research directions, and a conclusion is provided in section seven.

II. SMART CARD OWNERSHIP MODELS

Smart card ownership models relate to who owns smart cards and in this section, two contrary models are discussed.

A. Issuer Centre Smart Card Ownership Model

In the Issuer Centric Smart Card Ownership Model, organisations referred as card issuers (i.e., banks, telecoms, and transport, etc.) acquire smart cards from card manufacturers to support their smart card-based service architecture. The card issuers then distribute them to their individual customers, who then use them at service access points to request services eligible by the card, as shown in figure 1.

![Issuer Centric Smart Card Ownership Model](image)

Figure 1. Issuer Centric Smart Card Ownership Model

Although multi-application smart cards allow applications from different organisations to co-exist in a secure and reliable manner, the card issuer retains ownership of smart cards and control over which applications can be installed or deleted. The ownership issues along with branding, and varying business objectives has decelerated the multi-application smart card adoption [9].

The advantages of this approach are issuance control, security control, modification control, and communication and
feature control, whereas disadvantages can include card proliferation (a different card for each service), restriction of user choice, barriers to rapid service roll-out, and costs [10].

B. User Centric Smart Card Ownership Model

The NFC effectively enables a mobile phone to emulate a contactless smart card, and this has stirred up interest in the multiple applications on a smart card [2]. In retrospect of the ICOM and its possible role in the deceleration of multi-application smart card initiative, we proposed the User Centric Smart Card Ownership Model (UCOM) [8].

The UCOM enables a dynamic, ubiquitous and on-demand installation or deletion of an application, according to the cardholder’s requirements. In the UCOM, organisations develop their application and lease them to their customers on request. The lease of an application stipulates the security requirements and operational restrictions that a smart card has to satisfy, the policy that governs these aspects being referred to as the Application Lease Policy (ALP) [10].

A user acquires a User Centric Smart Card (UCSC) from a card manufacturer, as shown in figure 2. Initially, the smart card might not have any applications installed on it. The user then requests an application from an organisation referred to as a Service Provider (SP). An SP is similar to the card issuer in the ICOM; however, in the UCOM they do not provide smart cards. They only develop applications, and their registered customers can then install them on their smart cards.

Although technologies like Multos [11], Java Card [12] and GlobalPlatform [13] provide the functionality to install and delete applications as required, they are based on the ICOM. Therefore, most of these technologies fall short of satisfying the security and operational requirements of the UCOM [8]. The advantages of the UCOM could include: dynamic availability, faster application (service) roll-out, ease of management, and cost effectiveness. Possible disadvantages might be: increased complexity, unproven security, communication control, and feature interaction [10].

A. Overview of the Simulator Problem

The application development process for embedded devices relies on device simulators to try, test and debug applications. Device simulators are a useful tool for application developers, as they can verify their code before deploying onto a live device. The simulator problem refers to a possible scenario in which a malicious user could remotely install an application onto a smart card simulator. One thing to note is that the simulator problem is only related to the remote installation and not to the on-site installation. In remote installation, a smart card is not present at an SP’s site, and the application is downloaded over the Internet. In such a scenario, therefore, how could an SP be sure that their application is not installed on a simulated device?

It can be asserted that simulators are used in a number of different environments, especially mobile application development, and do not present a substantial security issue in the mobile application environment. Nevertheless, the nature of an application installed on a smart card is different than an application on a mobile phone. The smart card application might represent the identity of the user, along with serving as a security token to access some services (including financial services). The nature of the service or business environment that a smart card application deals with is substantially different than that of a mobile phone application.

In the ICOM, the simulator problem is not relevant, as applications are predominantly installed by the card issuer before the smart cards are issued to individual users. This stage in the smart card lifecycle is also referred as the pre-issuance stage. The GlobalPlatform specification provides the framework for application installation under the application provider’s control, after the smart cards are issued to customers. The GlobalPlatform defines a secure entity on the smart card referred to as the Card Manager, along with associated domains [13]. This requires symmetric keys in order to gain access to the domains (application domains) and install applications. The assumption in the ICOM is that malicious users cannot access or retrieve these keys. The basis of this assumption is on the tamper-resistance property of the smart card hardware that is certified by the Common Criteria (CC) evaluation [14].

In the UCOM, SPs might not have any off-card relationship with the smart card (or the card manufacturer). The UCOM-based smart cards are acquired directly by a user from a manufacturer of his or her choice. It is virtually unfeasible from an SP’s perspective though to have a trusted relationship with all possible smart card manufacturers. Another issue is that, even if an SP has an off-card relationship with a smart card manufacturer, the UCOM requires preservation of the user’s privacy. This requirement places stringent restrictions on a smart card to not contact its manufacturer. A possible solution could be for the smart card manufacturer to copy the trusted keys that it has established with all of the partner SPs to their smart cards. But this raises the issue of memory restrictions and the possible privileging of particular SPs. The UCOM should be a neutral framework, and a smart card should not prefer any particular SP or application. Therefore,
it is difficult to ascertain whether a smart card is what it claims to be and not a simulator. The issue is not only the problem of verifying the existence of a smart card, but also in validating that it is in a secure and reliable state. The capabilities of malicious users regarding the simulator problem are as follows:

1) Simulate the desired card environment on a computer.
2) Has the knowledge of the underlying platform, including smart card hardware and software.
3) Is capable of performing the side-channel analysis.
4) Can request for an application as an authentic user.

B. Scenarios of Simulator Based Attacks

In this section, we discuss possible scenarios of the simulator problem, which a malicious user can deploy.

1) Complete Simulation: In this configuration, a malicious user has a stand-alone simulator. He or she then tries to acquire an application from an SP to install on the simulator in order to conduct reverse engineering.

2) Partial Simulation: A malicious user uses the genuine smart card (hardware) and partially simulates the application storage and runtime environment, as shown in figure 3. The attacker’s aim is to provide assurance of the hardware presence to an SP before the application installation process begins, with the application then being redirected onto the simulator.

3) Simulation-in-the-Middle: The simulator intercepts the communication between an SP and a smart card. To the SP, it presents itself as a smart card and vice versa as shown in figure 4. In this scenario, a malicious user simply monitors the network traffic in order to install an application in their simulated environment that does not belong to them.

IV. PROPOSED FRAMEWORK

We open our discussion with a description of the smart card architecture and the main components of the proposed framework. Then, in section 4.2, the proposed protocol is described.

A. Smart Card Architecture

The validation and assurance mechanism that assures that a smart card is what it claims to be requires architectural changes, like the inclusion of a Trusted Platform Module [15] for smart cards, as illustrated in figure 5.

<table>
<thead>
<tr>
<th>Application A</th>
<th>Application B</th>
<th>Application C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime Environment</td>
<td>Virtual Machine</td>
<td>Native Code</td>
</tr>
<tr>
<td>Trusted Platform Module</td>
<td>Smart Card Hardware</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Smart Card Architecture in UCOM

1) Trusted Platform Module: We aim to investigate the complete architecture and functionality supported by a TPM on a smart card. Therefore, only the relevant aspects of the simulator problem are included in this paper.

A TPM on smart cards would basically provide assurance and validation of the platform. It stores a unique public key pair that is bonded to the smart card, along with the security evaluation certificate issued by the CC evaluation authority [16].

A TPM for smart cards does not have to be in the hardware. It can be software-based, and utilise the smart card’s cryptographic hardware (crypto co-processor). It is evident that the malicious user can also emulate the TPM, and the proposed framework takes this into account.

One point to note is that a general TPM implementation takes the role of a TPM as a static entity. However, the TPM in smart cards takes the role of an active entity that is involved in the different processes of the smart card. The final appraisal of the TPM for a smart card would not only include the common or traditional TPM functionality, but also the additional responsibilities, including the trusted domain manager, an independent on-card Controlling Authority (CA) and the Supplementary Security Domain. The last two roles are defined in the GlobalPlatform specification [6], but it does not mention TPM and we consider that a smart card TPM would be suitable to take on these roles.

C. Requirements for the Proposed Solution

Any proposed framework to mitigate the simulator problem should meet the following requirements:

1) Requiring minimum changes to smart card hardware.
2) It should be part of the application installation process without imposing too many constraints.
3) The solution should be robust, reliable and secure.
4) Any interruption to the execution of the framework should result in disabling the smart card.
5) It should preserve the privacy of the cardholder. A card manufacturer does not need to be involved, as this could lead to a privacy violation if the card manufacturer keeps the records of which applications are being installed onto their smart cards.
2) Common Criteria Certificate: The CC evaluation methodology certifies a smart card’s security mechanisms, which include cryptographic algorithms, application sharing and execution protection, and protection against Side-Channel attacks, etc. This was in contrast to the present procedure, in which at the end of the certification process, the CC certification authority issues an offline (paper-based) certificate. An electronic certificate [13] would be issued that is digitally signed by the CC certification authority and it would be stored on individual smart cards of the evaluated batch.

When an entity requests a TPM for a Smart Card Operating System (SCOS) or application validation, it would generate and sign a validation proof that consists of the attestation value of the requested component.

Smart card manufacturers and application developers would take their product to a specialised facility for evaluation, as illustrated in figure 6. The CC certification authority will then issue a cryptographic certificate after verification and validation of the product has taken place. In addition to this certificate, which would include the attestation (hash) value of the evaluated product, the CC certification body would also certify a manufacturer or SP’s signature key pairs.

![Figure 6. Illustration of Security Assurance and Validation Mechanism](image)

Manufacturers and SPs could then certify the unique signature key pair of each instance of their product, as illustrated in table 1. Therefore, each smart card or application would have a unique public key pair (for a digital signature), certified by their manufacturer. The public key, also referred as the "verification key", would act as a unique serial number for the product (smart card or application). In this paper, the private key is referred as the "validation key".

<table>
<thead>
<tr>
<th>Key Pair</th>
<th>Key Pair Owner</th>
<th>Certified By</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCC, SCC</td>
<td>CC National Authority</td>
<td>Self-certified</td>
</tr>
<tr>
<td>PCC-E, SCC-E</td>
<td>CC Evaluation Body</td>
<td>CC National Authority</td>
</tr>
<tr>
<td>PM, SM</td>
<td>Card Manufacturer</td>
<td>CC Evaluation Body</td>
</tr>
<tr>
<td>PTPM, STPM</td>
<td>Card TPM</td>
<td>Card Manufacturer</td>
</tr>
<tr>
<td>PSC, SSC</td>
<td>Card User</td>
<td>Card TPM</td>
</tr>
</tbody>
</table>

Table I

KEY AND CERTIFICATE HIERARCHY IN THE UCOM

In the list of keys mentioned above, only the smart card user key pair is used for encryption, whereas all of the other keys are signature key pairs. The smart card signature key pair is stored in the TPM, and could be described as the permanent TPM keys. Furthermore, when a user acquires a smart card, he or she would take ownership of the card. The ownership process includes the generation of a new public key pair (for encryption) that is bonded to the individual card-user. The public key pair would be for a particular user for a specific smart card that he or she owns. When smart card ownership is changed among users, a new public key pair would be generated, which would then be certified by the smart card TPM. In this paper, we call it the smart card key pair and certificate. Although it is related to the user, it is stored in the TPM’s secure storage, and the TPM could only be used after explicit permission of the smart card owner.

B. Anti-Simulator Protocol

The protocol is integrated into the UCOM’s application installation mechanism at two critical phases: requirement verification and localisation [10]. In the requirement verification phase, an SP would analyse the security mechanisms implemented by a smart card and ascertain whether it conforms to its ALP (section II.B). The localisation process deals with the registration of the application with different services on a smart card such as firewall mechanisms [17]. At this stage, sensitive data (i.e. cryptographic keys and the user’s information) are not part of the application, and it is downloaded or decrypted during the personalisation phase [10]. The protocol is illustrated in figure 8, and is described below.

1) Requirement Verification Phase: A secure connection is established between a smart card and an SP before the installation application is initiated [10]. During the requirement verification phase, the smart card requests that the SP lease its application. In response to this message, the SP would request the smart card’s TPM to provide assurance of the current state of the smart card environment.

On receiving this request, the TPM would initiate a self-test mechanism of different card security (hardware) components. If the result of the self-test is shows that the TPM is secure, the TPM would unlock the TPM signature key that is used to sign the responses to the SP during the protocol.

The TPM would then initiate the SCOS attestation process. In this process, the TPM generates the hash of the SCOS and concatenates it with the SP’s random number [18] to avoid replay attack, at which point the TPM signs the result with the validation key. The response message consists of the CC evaluation certificate, verification key and certificate and the signature over SCOS’s hash. At this point, the SP would try to verify the authenticity of the CC evaluation certificate sent by the smart card by retrieving the certificate from the certification authority website. If the SP does not trust the CC certification authority, it would traverse the certificate chain until it finds a CC authority that it trusts [16].

The reason for doing so is to accommodate the varying cross-country recognition of CC [14], and the possibility that smart cards and applications could be evaluated by different CC evaluation authorities [16]. In addition, during this process the SP would also ascertain whether the Evaluation Assurance Level (EAL) [16], to which the smart card is certified,
translates to the minimum requirement of the SP’s ALP. The hash value is then matched with the CC evaluation certificate, along with the verification of the validation signature. If the verification is successful, the encrypted application is downloaded to the smart card. The application is not decrypted until the second phase of the proposed protocol, which is executed in the localisation phase.

A session key (symmetric key) [19] is generated at the end of the requirement verification phase. The session key is then used to secure the application download; a process that is part of the “application download phase” [10], which does not fall within the scope of this paper.

2) Localisation Phase: In the application download phase, the application is transferred from the SP’s application server to the smart card. It is not active or installed onto the smart card though. The localisation phase would register the application with different services on the smart card. In addition, during this phase the SP requests the smart card’s TPM to perform the application verification. The TPM then generates the hash of the downloaded application, and would not only verify the result itself, but would also send the generated value to the SP.

On receipt of this message, the SP would compare the test results with the CC evaluation certificate. If successful, it would then allow decryption of the application and registration with different services on the smart card.

C. Protocol Messages Description

In the previous section, we gave a generic overview of the proposed anti-simulator protocol. In this section, an implementation of the proposed framework is listed that is later analysed using the Casper/FDR approach.

Before listing the structure of messages in the anti-simulator protocol, the following table describes the notations used in the protocol description.

The main aim of the protocol is to verify the smart card, ascertain its security conformance, prevent simulator attacks, and communicate the application to it without any third party being able to retrieve the downloaded application. The description of the protocol is as follows:

1) $SP \rightarrow SC : (SP|N_{SP}|Cert_{SP})$
After the SP receives the request from the cardholder to install their application onto his or her smart card, the SP would send the signature key certificate, along with a random number. The message is routed to the smart card TPM and would then communicate with the SP for the rest of the protocol.

2) $T \rightarrow SP : E_{PK_{SP}}(SC|N_{SP}|N_{T}|g^{nx}|g|p)|Cert_{SC}|Cert_{T}$
The smart card TPM would generate a random number that is encrypted along with the identities of the SC and SP, and global parameters $g$, $p$ and $g^{nx}\mod p$ for key
V. ANALYSIS OF THE PROTOCOL

A. Formal Proof of the Protocol

The proposed protocol was tested by the Casper/FDR approach [20]. In this approach, the Casper compiler [21] is used to translate the high-level description of the protocol into the processing algebra of Communicating Sequential Processes (CSP) [22]. The CSP description is then analysed by a machine-verifying tool known as the Failure-Divergence Refinement (FDR) model checker [23].

The CSP-programme consists of processes that represent agents involved in the protocol, running in parallel with an intruder. The basic assumption in the CSP-programme is that all the agents taking part in the protocol could be dishonest. The FDR verifies the CSP description and tries to find a possible means of attack. If found, then it would be translated by the Casper compiler back into a human-readable format.

The Casper-FDR script for the proposed protocol is listed in Appendix A. In this section, we will discuss the assumptions that were made in the implementation of this protocol.

The assumptions regarding the network are: i) A Common Criteria entity is a trusted entity and certificates issued by it could not be faked. The CC evaluation certifies the security level that a product (i.e., smart cards and applications) satisfies; and ii) a smart card is a temper-resistant device. Adequate security measures are implemented to avoid side-channel attacks [24] and certified by the CC evaluation. The intruder capability in the network is listed as below:

1) An intruder can masquerade as a smart card or application identity in the network.
2) An intruder could not masquerade as a Common Criteria evaluation authority in the network. For this reason, therefore, we did not implement the CC evaluation authority in the Casper-FDR script.
3) An intruder could masquerade as a smart card TPM, but could not have access to the TPM keys (because of temper-resistant hardware).
4) It can read the messages transmitted on the network.
5) An intruder is not able to influence the internal processes of an agent in the network (with the exception of itself).
6) An intruder is not able to fake cryptographic certificates or hash values.

The security specification for which the Casper-FDR tool would evaluate the network is listed under the specification section of the Casper script in Appendix A:

1) The session key generated by an SP and a TPM is secure. Both entities have a mutual agreement on the session key.
2) The SP has an assurance that the protocol running is fresh and that the TPM is actively taking part in the protocol, and vice versa.

The formal proof can only provide assurance that the protocol running was secure and that no malicious entity was able to retrieve any sensitive information by listening to the communication channel. If we do not assume that the smart card is tamper-resistant, then a malicious user could retrieve the signature and encryption key pair of a smart card and use that information to simulate the environment.

B. Analytical Proof of the Protocol

The objective of the protocol is to prevent a malicious user from acquiring an application and installing it in his or her simulated smart card environment. To achieve this, we incorporated the tamper-resistant property of smart cards, TPM, the CC evaluation and the symmetric/asymmetric cryptography.

The role of the TPM is to provide assurance in the smart card environment (both SCOS and hardware). It generates the hash values of a SCOS and installed applications on a smart card to verify that their current state is as it was at the time of the CC evaluation. In addition, the TPM also stores the TPM signature key pair and smart card public key pair securely (for encryption). The CC evaluates the smart card security measures and issues a CC digital certificate that certifies the
TPM signature key pair. This certificate also contains the attested digest value of the SCOS.

To conduct a simulator attack, a malicious user has to gain access to the secure storage of the TPM and retrieve the signature key pair. As the proposed protocol is based on the security assurance provided by the TPM and attested by the TPM signature key pair, if the malicious user got the CC certified TPM signature key pair, he or she can perform a full or partial simulation; otherwise it is technically difficult, because the SP would detect the discrepancies in the cryptographic certificate chain. If the SP does not ascertain the platform assurance from the cryptographic certificate that it is being issued by a trustworthy CC evaluation authority, it might opt not to lease the application.

Finally, in the simulation-in-the-middle attack, the purpose is to download the application onto a simulator and then try to work out the cryptographic key(s) that are used to encrypt the application for transfer over an insecure network. The proposed framework does not avoid this attack, as it would always be possible to tap into the communication between a smart card and SP. However, the attack relies on retrieving the encryption key(s) that is used to encrypt the downloaded application. This could be achieved either by the side-channel analysis or by a cryptanalysis of the algorithm. As the application installation process uses symmetric keys to encrypt the application before transferring it, the possibility of retrieving the symmetric key is real. However, side-channel attacks require an extensive number of executions of an algorithm with a single key. Therefore, using multiple keys to encrypt different sections of the application could make it difficult for a malicious user to retrieve the key, either through an exhaustive search (for the key) or by the side-channel analysis.

VI. FUTURE RESEARCH DIRECTIONS

What follows is a short list of research questions that require in-depth analyses.

1) Hardware self-test mechanism: This mechanism should be robust, secure and rely on varied parameters. The results of the test should be consistent, capable of being regenerated, and comparable.

2) Application level modularity: This specifies the application (code) security partitioning in which each section of the code has a different security level and, depending upon their security level at the execution time, a relevant security policy is applied.

3) Ownership transfer mechanism: This concerns how the ownership of a smart card could be changed from one owner to another. This functionality would enable cardholders to sell or acquire smart cards as desired.

VII. CONCLUSION

In this paper, we discussed the User Centric Smart Card Ownership Model (UCOM) and how it gives a choice of applications on smart cards to their cardholders. The UCOM introduces unique security problems, one of which is the simulator problem whereby a malicious user can manage to download an application to a smart card simulated environment. This threat not only enables the reverse engineering of the application (i.e. intellectual property theft) but also the retrieval of sensitive information (e.g., personal data, cryptographic keys). We described different scenarios that a malicious user could deploy and then proposed a framework that could prevent them. The framework relies on the TPM and on-card CC certificates that are not implemented on smart cards. Therefore, they require an in-depth analysis to define their structure and how they will interact with on-card and off-card entities.

REFERENCES

APPENDIX

Casper/FDR Script

# Free variables
datatype Field = Gen|Exp(Field, Num) unwinding 2
halfkeySP, halfkeyTPM, sessionKey : Field
SP, TPM : Agent
ns, nt, nm, scos, app : None
s, t : Num
VKey : Agent -> PublicKey
SKey : Agent -> SecretKey
EKey : Agent -> PublicKey
InverseKeys = (sessionKey, sessionKey), (VKey, SKey), (EKey, DKey), (Exp, Exp), (Gen, Gen)

# Protocol description
0. -> SP : TPM
1. SP -> TPM : SP, VKey(SP)
2. TPM -> SP : {TPM, SP, nt}{VKey(SP)}
2a. TPM -> SP : {Exp(Gen, t)@%halfkeyTPM} {VKey(SP)}
   < sessionKey := Exp(halfkeyTPM, s) >
3. SP -> TPM : {SP, TPM, ns}{EKey(TPM)}
3a. SP -> TPM : {Exp(Gen, s)@%halfkeySP} {EKey(TPM)}
   < sessionKey := Exp(halfkeySP, t) >
4. TPM -> SP : {TPM, SP, {scos(+ ns)}{SKey(TPM)} } {sessionKey}
5. SP -> TPM : {SP, TPM, nt}{sessionKey}
6. TPM -> SP : {TPM, SP, {app(+ ns)}SKey(TPM) } {sessionKey}

# Actual variables
SerPro, TruPlaMan, MAppl : Agent
Nsp, Ntsp, Nm : None
SCOS, APP : None
S, T, M : Num
S COperingSys, SApplication : None