A Secure and Trusted Channel Protocol for UAVs Fleets

Raja Naeem Akram, Konstantinos Markantonakis, Keith Mayes,
Pierre-François Bonnefoi, Amina Cherif, Damien Sauveron, Serge Chaumette

▶ To cite this version:

Raja Naeem Akram, Konstantinos Markantonakis, Keith Mayes, Pierre-François Bonnefoi, Amina Cherif, et al.. A Secure and Trusted Channel Protocol for UAVs Fleets. 11th IFIP International Conference on Information Security Theory and Practice (WISTP), Sep 2017, Heraklion, Greece. pp.3-24, 10.1007/978-3-319-93524-9_1 . hal-01875525

HAL Id: hal-01875525

https://inria.hal.science/hal-01875525

Submitted on 17 Sep 2018

A Secure and Trusted Channel Protocol for UAVs Fleets

Raja Naeem Akram[1], Konstantinos Markantonakis[1], Keith Mayes[1],
Pierre-François Bonnefoi[2], Amina Cherif[2,4],
Damien Sauveron[2,3], and Serge Chaumette[3]

[1] Information Security Group Smart Card Centre, Royal Holloway, University of London, Egham, United Kingdom
[2] XLIM (UMR CNRS 7252 / Université de Limoges), MathIS. Limoges, France
[3] LaBRI (UMR CNRS 5800 / Université de Bordeaux), Talence, France
[4] LARI (Université Mouloud Mammeri de Tizi-Ouzou), Tizi-Ouzou, Algeria
Email: {r.n.akram, k.markantonakis, keith.mayes}@rhul.ac.uk,
{pierre-francois.bonnefoi, damien.sauveron}@unilim.fr,
amina.cherif@etu.unilim.fr, serge.chaumette@labri.fr

Abstract. Fleets of UAVs will be deployed in near future in reliability and safety critical applications (*e.g.* for smart cities). To satisfy the stringent level of criticality, each UAV in the fleet must trust the other UAVs with which it communicates to get assurance of the trustworthiness in information received and to be sure not to disclose information to an unauthorized party. In addition, to be protected against an attacker willing to eavesdrop and/or modify the exchanged data, the communication channel needs to be secured, *i.e.* it has to provide confidentiality and integrity of exchanges. The work presented here is based on our previous research which concluded that it is required that each UAV includes a Secure Element (which we called ARFSSD standing for Active Radio Frequency Smart Secure Device) to withstand an adversary with a high attack potential. In this paper, we propose a secure and trusted channel protocol that satisfies the stated security and operational requirements for a UAV-to-UAV communication protocol. This protocol supports three main objectives: 1) it provides the assurance that all communicating entities can trust each other and can trust their internal (secure) software and hardware states; 2) it establishes a fair key exchange process between all communicating entities so as to provide a secure channel; 3) it is efficient for both the initial start-up of the network and when resuming a session after a cold and/or warm restart of a UAV. The proposed protocol is formally verified using CasperFDR and AVISPA.

1 Introduction

There are increasing number of application-areas that consider the usage of Unmanned Aerial Vehicles (UAVs), and specially of fleets of UAVs. It is thus of major importance to propose security mechanisms to provide strong guarantees in terms of reliability, safety, privacy-protection and security. Regardless of the field of applications, whether military or civil, fleets of UAVs of course have to operate as planed and they thus should resist an adversary trying to tamper its reliability and safety, for instance by conducting attacks on the communication network between the UAVs to make them crash or misbehave. In addition, if in most civilian applications privacy-protection of end-users collected data is a required property for them to be accepted or certified-for-use, in military applications security (in term of confidentiality and integrity) of on-board data (*i.e.* the collected data and also the pre-loaded data) is a mandatory requirement.

As shown in [1] which dealt with adversary models for UAVs fleets, each UAV must be equipped with a Secure Element (SE) so as to withstand an adversary with a high attack potential. In this paper, based on presence of such SEs, we propose a secure and trusted channel protocol that satisfies the stated security and operational requirements for a UAV-to-UAV communication protocol.

1.1 Contribution

In this paper, our main goals are to propose a secure and trusted channel protocol for fleets of UAVs, and to compare its security and performance with similar protocols.

The salient contributions of this paper are the following:

1. proposed a Secure and Trusted Channel Protocol (STCP) to establish a secure channel between the communicating UAVs and to provide security assurance that each UAV is in the secure and trusted state;
2. defined comparison criteria for secure channel protocols along with the related security analysis;
3. validated the proposed protocol with a mechanical formal tools: CasperFDR and AVISPA.

1.2 Structure of the Paper

Section 2 briefly presents the domain of UAVs fleets, the associated security issues, how by providing the assurance to communication partners that the nodes are secure and trustworthy SEs can help to secure the fleet and the rational for a STCP. Section 3 discusses the existing work carried out in terms of UAV applications and secure channel protocols from a traditional computer security perspective. Section 4 discusses the proposed security comparison criteria and the proposed protocol for a SE-equipped UAVs fleet. In section 5, before to formally analyze the proposed protocol using CasperFDR and AVISPA, we first compare it with different secure channel protocols of the related work based on the security comparison criteria previously defined. Finally in section 6 we present future research directions and conclude the paper.

2 UAVs Fleet and Rationale for a STCP

A fleet of UAVs is composed of a set of small and light UAVs flying in swarm formation and collaborating together to achieve the entrusted mission. Each

UAV is equipped with sensors which might be different of those of the other UAVs of the fleet. Additionally, for reliability reasons, there might be some redundant sensors. Since UAVs fleets can cover large geographic areas they are a possible replacement for regular big and expensive drones used in the past both in military and civilian applications. For instance, in the civilian applications such fleets of UAVs can be used for monitoring forest fires, searching missing people in avalanches, etc. As illustrated Fig. 1, to collaborate together, UAVs have to communicate. However if the recipient, application (3) running on UAV C, is not in the scope of the sender, the application (1) running on UAV A, the message must be routed like in a Mobile Ad hoc Network (MANet) by intermediary nodes (UAVs), here UAV B.

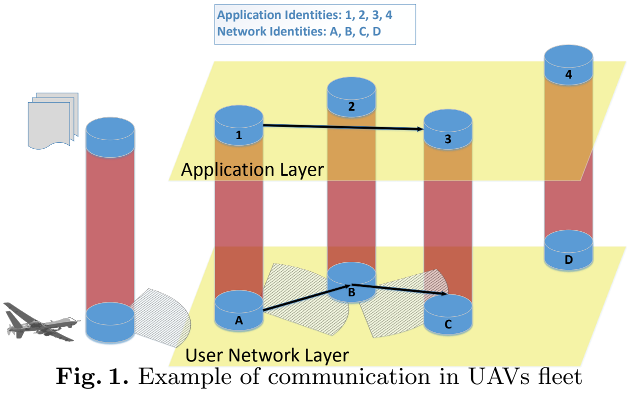

Application Identities: 1, 2, 3, 4
Network Identities: A, B, C, D
4
2
1
3
Application Layer
D
B
A
C
User Network Layer

Fig. 1. Example of communication in UAVs fleet

2.1 Assets to Protect, Adversary Model and SE

Depending on the applications in which such a fleet is deployed and of the chosen adversary model, the assets to protect differ. In terms of information security, assets are the valuable data (here, data also includes the software application: intellectual property) of the owner of the UAVs fleet but are also valuable for attackers. Fig. 2 depicts the general targets of an attacker on an individual UAV. There are different kinds of assets that might interest them, among which:

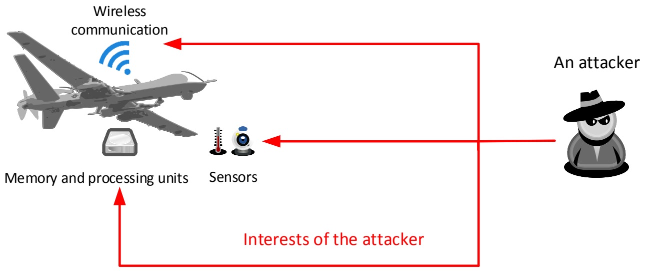


Fig. 2. Attacker Interests

- pre-loaded data: *i.e.* flight-plan of the mission, cryptographic keys used to secure the communications, code of the applications running on the UAV, etc.
- collected data: *i.e.* photos, coordinates of points of interest (enemies or allies in case of a military application), etc.
- communication-related data: *i.e.* routing tables, session keys, etc.

In our previous work [1], we have considered a strong adversary model with a high attack potential, *e.g.* the adversary has the capabilities and knowledge to capture a UAV in a functional state, to perform side-channel attacks or fault injections or other physical, software or combined attacks in order to gain access to (or to modify for his/her profit) some secret data (*e.g.* cryptographic keys), software or hardware. We have proposed a rationale which concluded that such a strong adversary model made sense, especially in the context of military usage of UAVs fleets, since the opponent can be a government-controlled organization capable of performing forensic analysis or attacks of the UAVs. Based on these conclusions we have derived the security and functional requirements and we have analyzed which one among several existing Secure Elements (*e.g.* Trusted Platform Module, active RFID, smart card) might be added to individual UAVs of the fleet to enhance the security up to the target assurance level.

Unfortunately none of them fulfilled all the criteria and we have proposed to use the one satisfying most of the requirements, i.e. the UCOM smart card (a smart card based on the User-Centric Ownership Model [2]), and to supplement it with the only missing feature which was the long range RF communication capability. This new SE, that we have named ARFSSD (Active Radio Frequency Smart Secure Device), is depicted Fig. 3.

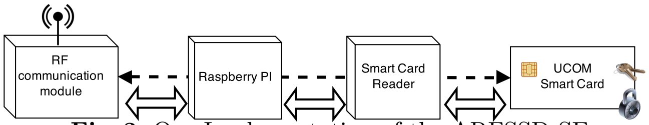


Fig. 3. Our Implementation of the ARFSSD SE

We do not intend to detail this SE, still we have to explain how it is used to equip each UAV and what are its security features since we use them to propose in this paper the missing secure and trusted channel protocol required to secure UAVs fleets.

2.2 SE Usage and its Security Features

As illustrated Fig. 4, a fleet of SE-equipped UAVs enables to build a control network layer between the SEs to provide high level of security for any exchanges in the upper network layers. This control network layer makes it possible to ensure that intermediary UAVs of the same fleet will not have access to the routed information (not even to the destination address if this is required for some privacy reasons). To ensure this kind of security properties, the destination address can be ciphered and the deciphering process can be done in the secure element which will decide if the message is for its own UAV or if it must be forwarded to another UAV of the fleet. Obviously the payload is also ciphered.

In addition, the SEs can also offer security services (like cryptography, secure storage and processing capabilities) to the application layer. In a UAVs fleet composed of UAVs belonging to several distinct owners or even in a UAVs fleet shared by several owners and running different applications. It might be possible to consider that some intermediary UAVs can be selfish. The presence of SEs enables to solve such an issue since the UAV itself is not aware of the routing decisions and some collaborative mechanisms can be also added (for instance based on reputation, or on retribution).

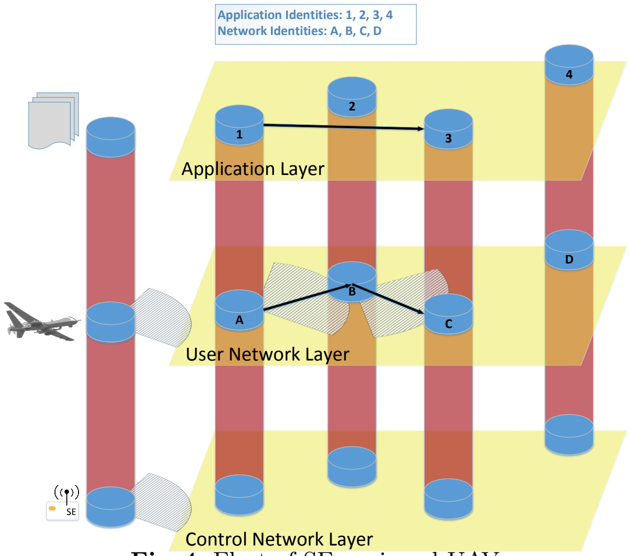
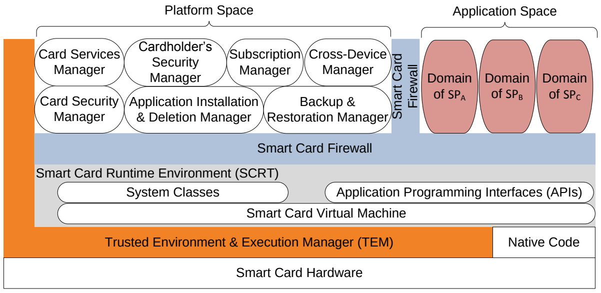


Fig. 4. Fleet of SE-equipped UAVs

In short, the SE depicted Fig. 4 can be defined as a long range RF-enabled UCOM smart card. One of the interests to have an SE equipped with long range communication capability is that the SEs altogether form an overlay network for specific control operations. This control network is parallel to the communication network, called User Network, that already exists between UAVs. In this paper we will not detail the characteristics of the RF interface of the SEs (which should be different of those of UAV: *e.g.* the interface can use a different RF spectrum; the bandwidth can be smaller but the radio coverage can be larger). Our objective is to propose a protocol to establish a secure and trusted channel between these SEs which are the roots of trust and security for a UAVs fleet architecture withstanding an adversary with a high attack potential.

The overall architecture of a UCOM based smart card [2] is illustrated in Fig. 5. Basically it is a multiapplication smart card supplemented with new components. The most important for our protocol is the TEM (Trusted Environment & Execution Manager) which is represented as a layer between the smart card hardware and the runtime environment. This illustration provides a schematic view of the architecture and does not imply that all communications between the runtime environment and the hardware goes through the TEM.



Platform Space
Application Space
Card Services Manager
Cardholder's Security Manager
Subscription Manager
Cross-Device Manager
Card Security Manager
Application Installation & Deletion Manager
Backup & Restoration Manager
Smart Card Firewall
Domain of SP_A
Domain of SP_B
Domain of SP_C
Smart Card Firewall
Smart Card Runtime Environment (SCRT)
System Classes
Application Programming Interfaces (APIs)
Smart Card Virtual Machine
Trusted Environment & Execution Manager (TEM)
Native Code
Smart Card Hardware

Fig. 5. UCOM Smart Card Architecture

As depicted Fig. 6, the TEM is supporting the Trusted Computing Base (TCB) by providing several similar functionalities (usually present on TPM chip) which are useful for our protocol. The attestation handler and the self-test manager are the main building blocks that can be used to ensure a kind of trusted boot. More details on how the TEM is implemented and what are the roles of the different components are available in [3]. Basically, the self-test manager and the attestation handler can provide the assurance that the current hardware and software state is secure as it was at the time of third party evaluation. This attestation process, called Platform State Verification/Assurance, can be requested by an internal entity on the card (*e.g.* an application) or by a remote party.

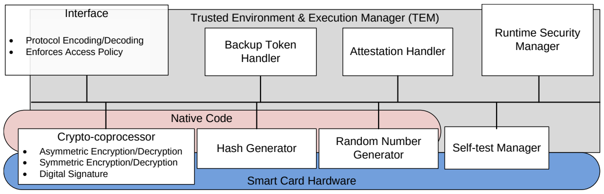


Fig. 6. Trusted Platform Module for Smart Card Architecture

These functionalities of the TEM will be used in the secure and trusted channel protocol proposed in section 4.

2.3 Rationale for a STCP

A Secure Channel Protocol (SCP) by definition provides either or both of entity authentication and key exchange between communicating parties (end points). A

SCP preserves the confidentiality and integrity of the messages on the considered channel but not at the end points.

Nevertheless, there can be implicit assurance in the integrity and security of the end points as described by ETSI TS 102 412 [4] in the domain of the smart card industry. This document states that the smart card is a secure end point under the assumption that it is a tamper-resistant device. This type of assurance can be extrapolated to other devices that are implicitly trusted because of offline business relationships or because of a property of the device itself.

However, for a critical system like a fleet of UAVs it is not just implicit trust that is required but also explicit trust validation, to counter any potential threat. The explicit trust assurance should be provided by the UAVs (here the SE-equipped UAVs) that are participating in the communication of the UAVs fleet. This would build in an assurance that only secure and trusted devices (explicitly trusted devices with per-protocol run assurance) will participate in the UAVs fleet, potentially countering physically altered devices and/or re-introduction of a decommissioned device.

A trusted channel is a secure channel that is cryptographically bounded to the current state of the communicating parties [5]. This state can be a hardware and/or a software configuration, and ideally it requires a trustworthy component to validate that it is effectively as claimed. Such a component, in most instances, is a TPM [6] as demonstrated in [7]–[9]

Even though in a fleet of UAVs, individual devices will have prior relationships with each other (at least through the owners of the UAVs – in case of a multi-owners UAVs fleet, prior relationship must be defined as seen in section 4.4.), when establishing a secure channel, individual devices should still ensure that they are not only communicating with an authenticated device but also that the current state of this device is secure.

3 Related Work

In this section, we review the existing work in two different areas: UAVs fleet and Secure Channel Protocols (SCPs).

3.1 Related Work on Security Concerns of UAVs Fleets

This section describes different work related to UAV secure communications.

In [10], the authors proposed a secure channel protocol between individual UAVs and a ground station (GS). When a data communication with GS is possible, the UAVs send their collected data. To avoid that an attacker can retrieve plaintext data in case a UAV is captured, each block of data is ciphered with a one-time key generated by a key stream. This protocol is efficient to protect the confidentiality of sensed data. To protect against the forgery of messages, a tamper-resistant element (i.e. an SE) is required. In [11], the authors also proposed a protocol to secure communication between individual UAVs and a GS along with ensuring that an illegitimate access to sensed data is not easily available to an attacker. However the proposal is not as efficient as [10] due to the use Off-the-Record messaging to provide strong properties (*e.g.* deniability that does not able a GS to prove to other parties that a specific message was received from a specific UAV – this property is useful to protect journalism sourcing) which are useless in the context of UAVs fleet.

In [12], the authors presented their HAMSTER (HeAlthy, Mobility and Security based data communication archiTEctuRe) solution for unmanned vehicles. However the paper rather describes a security framework and cryptographic schemes than secure channel protocols. In the paper, they proposed some benchmark of ECC-based schemes (instead of RSA since ECC is more suitable for constrained devices) the performance of which was measured on a PC. However, the security of the private keys are not addressed.

In [13], the authors proposed SUAP, a secure reactive routing protocol the main cons of which is that it does not consider an adversary with a high attack potential. However, it is efficient to detect and prevent routing (*e.g.* wormhole, blackhole) attacks.

In [14], the authors proposed a secure communication protocol between UAVs and smart objects. If this is not exactly the same objective as that of our proposal, this paper was interesting since it took into account the capture of a UAV. The proposed protocol was based on efficient Certificateless Signcryption Tag Key Encapsulation Mechanism using ECC. However the solution does neither address smart objects capture nor the peer-to-peer communication mode of the UAVs fleet.

3.2 Related Work on Secure Channel Protocols

In this section, we restrict the discussion to the protocols that are proposed for general-purpose computing environments or to those that are used as references for comparison in the discussions to come.

The concept of trusted channel protocol was proposed by Gasmi et al. [5] along with the adaptation of the TLS protocol [15]. Later Armknecht et al. [8] proposed another adaptation of OpenSSL to accommodate the concept of trusted channels; similarly, Zhou and Zhang [7] also proposed a SSL-based trusted channel protocol.

In section 5.2, we will compare the proposed STCP with the existing protocols. These protocols include the Station-to-Station (STS) protocol [16], the Aziz-Diffie (AD) protocol [17], the ASPeCT protocol [18], Just-Fast-Keying (JFK) [19], trusted TLS (T2LS) [5], GlobalPlatform SCP81 [20], the Markantonakis-Mayes (MM) protocol [21], and the Sirett-Mayes (SM) protocol [22].

This selection of protocols is intentionally broad so as to include well-established protocols like STS, AD and JFK. We also include the ASPeCT protocol, which is designed specifically for value-added services of mobile networks. Similar to our proposal where we require trust assurance during the protocol run, T2LS meets this as it provides trust assurance, whereas other protocols like SCP81, SM, and MM are specific to smart cards and are representative for embedded low-power devices. In addition, we have included the secure and trusted channel protocol, P-STCP [9], which is designed for resource-restricted and security-sensitive environments, and has some similar design requirements to those of the proposed protocol.

4 Secure and Trusted Channel Protocol

In this section, we begin the discussion with the security comparison criteria, followed by the protocol notation, the pre-setup and then the actual protocol proposal. This section concludes with a discussion of how the secure channel is re-established if one of the devices is restarted or resets the protocol.

4.1 Security Comparison Criteria

For a protocol to support the UAVs fleet, it should meet, at minimum, the security and operational requirements listed below:

G1) ***Mutual Entity Authentication*****:** All nodes in the network should be able to authenticate to each other to avoid masquerading by a malicious entity.

G2) **Asymmetric Architecture:** Exchange of certified public keys between the entities to facilitate the key generation and entity authentication process must be provided.

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 prevent one party from 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 against replay attacks.

G6) **Mutual *Key Confirmation*:** Communicating parties will provide implicit or 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 the session key of a particular protocol run, it should not enable him/her to retrieve long-term secrets (*private keys*) or *session keys* (future and past).

G8) **Unknown *Key* Share Resilience:** In the event of an unknown key share attack, an entity $\mathcal{X}$ believes that it has shared a key with $\mathcal{Y}$, where the entity $\mathcal{Y}$ mistakenly believes that it has shared the key with entity $\mathcal{Z} \neq \mathcal{X}$. The proposed protocols should adequately protect against this attack.

G9) ***Key*** **Compromise Impersonation (KCI) Resilience:** If a malicious user retrieves the long-term key of an entity $\mathcal{Y}$, it will enable him/her to impersonate $\mathcal{Y}$. Nevertheless, key compromise should not enable him/her to impersonate other entities to $\mathcal{Y}$ [23].

G10) ***Perfect Forward Secrecy*****:** If the long-term keys of the communicating entities are compromised, this will 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 a protocol run with each other.

G12) **Partial Chosen Key (PCK) Attack Resilience:** Protocols that claim to provide joint key control are susceptible to this type of attack [24]. In this type of 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 (*i.e.* it is able to generate a partially weak key). However, this attack depends upon the computational capabilities of the second entity. Therefore, the proposed protocols should adequately prevent PCK attack.

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.

G14) **Denial-of-Service (DoS) Prevention:** The protocol should not require the individual nodes to allocate a large set of resources to the extent that it might contribute to a DoS attack.

G15) **Privacy:** A third party should not be able to know the identities of the SE-equipped UAVs.

For a formal definition of the terms (italicized) used in the above list, the reader is referred to [25]. The requirements listed above are later used as a point of reference to compare the selected protocols in Table 3.

4.2 Protocol Notation

The notations used in the protocol description are listed in Table 1.

4.3 Pre-Protocol Setup

The proposed protocol requires certain pre-protocol setup operations as listed below:

1. Each UAV that is part of the fleet has an ARFSSD, so called SE for concision reasons.
2. Each SE in the fleet is pre-configured with the signature verification keys of the owners of its partners (*i.e.* public keys of the owners of SE-equipped UAVs – note that all UAVs can have the same owner) to be able to verify the signature verification key of each SE contained in C_{SE}. Each owner willing his/her UAVs to take part of the fleet has to certify the SEs public key with his/her signature key and he/she has to provide his/her public key to the other owners willing to use his/her UAVs.
3. Each SE is also pre-configured with the signature verification keys of the certification body which has assessed the security of the SE and its TEM to be able to verify the signature verification key of each SE contained in $C_{TEM_{SE}}$. This certification of TEM public key is part of the UCOM smart card manufacturing process [27].

Table 1. Notations used in protocol description.

$SE1$	: Denotes an ARFSSD '1'.
$SE2$	: Denotes an ARFSSD '2'.
$A \rightarrow B$	: Message sent by an entity A to an entity B.
TEM_X	: Denotes the TEM of an entity X
X_i	: Represents the identity of an entity X.
g^{r_X}	: Diffie-Hellman exponential generated by an entity X.
C_X	: Signature key certificate of an entity X.
N_X	: Random number generated by an entity X.
$X\|Y$	: Represents the concatenation of the data items X, Y in the given order.
$[M]_{K_a}^{K_e}$	: Message M is 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.
$Sign_X(Z)$	: Signature generated on data Z by the entity X using a signature algorithm [26].
$H(Z)$	: 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 one of the communication entities. It indicates the session information and facilitates protection against DoS attacks along with (possibly) providing the protocol session resumption facility.
VR_{A-B}	: Validation request sent by entity A to entity B. In response entity B provides a security and reliability assurance to entity A.
SAS_{A-B}	: Security assurance generation by entity A that provides trust validation to the requesting entity B.

4. Each SE is also pre-configured with the security assurance values for the trusted and secure state of its communication partners.

One additional interest of our proposal is to support a multi-owner fleet of UAVs (UAVs can be owned by different parties). It is also possible to filter the UAVs authorized from "less" trusted owners by adding the SE identities attached to trusted UAVs.

4.4 Proposed Protocol

The messages of the protocol are listed in Table 2 and are described below.

Message 1 The SE1 generates a random number N_{SE1} and computes the Diffie-Hellman exponential $g^{r_{SE1}}$. The "$H(g^{r_{SE1}}\|N_{SE1}\|SE1_i\|SE2_i)$" serves as a session cookie "S_{Cookie}", and it is appended to each subsequent message sent by both devices. It indicates the session information, facilitates protection against DoS attacks and provides the protocol session resumption facility, which is required if a protocol run is interrupted before it successfully concludes. Finally, SE1 will request SE2 to provide assurance of its current state.

Table 2. Secure and Trusted Channel Protocol (STCP).

$$\begin{array}{rl}
1.\ SE1 \rightarrow SE2 : & SE1_i \| SE2_i \| N_{SE1} \| g^{r_{SE1}} \| VR_{SE1-SE2} \| S_{Cookie} \\
2.\ SE2 \rightarrow SE1 : & SE2_i \| SE1_i \| N_{SE2} \| g^{r_{SE2}} \| [Sign_{SE2}(SE2-Data) \| Sign_{TEM_{SE2}}(SE2-Validation) \| \\
: & C_{SE2} \| C_{TEM_{SE2}}]^{K_e}_{K_a} \| VR_{SE2-SE1} \| S_{Cookie} \\
: & SE2-Data = H(SE2_i \| SE1_i \| g^{r_{SE1}} \| g^{r_{SE2}} \| N_{SE1} \| N_{SE2}) \\
: & SE2-Validation = SAS_{SE2-SE1} \| N_{SE1} \| N_{SE2} \\
3.\ SE1 \rightarrow SE2 : & [Sign_{SE1}(SE1-Data) \| Sign_{TEM_{SE1}}(SE1-Validation) \| C_{SE1} \| C_{TEM_{SE1}}]^{K_e}_{K_a} \| S_{Cookie} \\
: & SE1-Data = H(SE1_i \| SE2_i \| g^{r_{SE2}} \| g^{r_{SE1}} \| N_{SE2} \| N_{SE1}) \\
: & SE1-Validation = SAS_{SE1-SE2} \| N_{SE2} \| N_{SE1}
\end{array}$$

Message 2 In response, SE2 generates a random number, and a Diffie-Hellman exponential $g^{r_{SE2}}$. It can then calculate the $k_{DH} = (g^{r_{SE1}})^{r_{SE2}}\ (mod\ n)$ which will be the a shared secret from which the rest of the keys will be generated. The encryption key is generated as $K_e = H_{k_{DH}}(N_{SE1} \| N_{SE2} \| ''1'')$ and a MAC key as $K_a = H_{k_{DH}}(N_{SE1} \| N_{SE2} \| ''2'')$.

Subsequently, the TEM generates a state validation message signed by the TEM private key represented in the protocol as "$Sign_{TEM_{SE2}}(SE2-Validation)$". SE2 will also request SE1 to provide assurance of its current state.

On receipt of this message, SE1 will first generate the session keys. It will then verify SE2's signature and validation proof generated by the TEM of SE2 after having verified that C_{SE2} and $C_{TEM_{SE2}}$ are genuine. As the signature key belongs to the TEM of SE2, an attacker cannot masquerade this signature. By verifying the signature, SE1 can ascertain the current state is measured by the TEM of SE2. Now SE1 can verify whether the security assurance value represents a trusted and secure state (or not) because since our pre-protocol setup, SE1 would have the security assurance value of a trusted and secure state of SE2.

Furthermore, SE1 will check the values of the Diffie-Hellman exponentials (*i.e.* $g^{r_{SE1}}$ and $g^{r_{SE2}}$) and of the generated random numbers to avoid man-in-the-middle and replay attacks.

Message 3 SE1 will then generate a message similar to message 2, a signature by SE1 and trust validation proof generated by its TEM.

On receipt of the message, SE2 will verify the two certificates, the trust validation proof and generate keys. It will also check the values of the Diffie-Hellman exponentials and of the generated random numbers to avoid man-in-the-middle and replay attacks.

4.5 Post-Protocol Process

The shared material generated from the Diffie-Hellman exponential can be used to generate more keys than just the session encryption and MAC keys of the protocol. If this is not desirable then the session encryption and MAC keys can be saved as master session keys.

4.6 Protocol Resumption

The protocol is run the first time that two SE-equipped UAVs of the fleet have to communicate together. The session cookie, S_{Cookie} is used to facilitate the session resumption subsequent exchanges. However, based on a predefined policy (*e.g.* based on the elapsed time since the first protocol run), the SE might require the protocol to be executed again to refresh the master session keys.

5 Protocol Evaluation

In this section, we first discuss the information analysis of the protocols, and then compare different protocols with our proposal based on the comparison criteria defined above. Finally, we provide a formal analysis using CasperFDR and AVISPA.

5.1 Brief Informal Analysis

Throughout this section, we refer to the protocol comparison criteria of section 4.1 by their respective numbers as listed in the same section.

During the proposed protocol, in messages 2 and 3 the communicating entities authenticate each other, which satisfies G1. Similarly, for G2, all communicating entities have exchanged cryptographic certificates to facilitate an authentication and trust validation proof (generated and signed by the TEM) before the SE-equipped UAVs are deployed (pre-deployment configuration).

The proposed protocol satisfies requirements G3, G4, G5 and G12 by first requiring SE1 and SE2 to generate the Diffie-Hellman exponential; computational cost is thus equal on both sides. Similarly, exponential generation also assures that both devices will have equal input to the key generation process. Messages 2 and 3 are encrypted using the keys generated during the protocol execution, thus providing mutual key confirmation (satisfying G6).

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

Furthermore, to provide unknown key share resilience (G8), the proposed protocol 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 which it has shared the key.

The protocol 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 device. Therefore, if $\mathcal{A}$ has knowledge of the signature key of a device, it can only masquerade this particular device to other devices but not others to it.

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

Goals	Protocols										
	STS	AD	ASPeCT	JFK	T2LS	SCP81	MM	SM	P-STCP	SSH	SSL	Proposed Protocol
G1.	*	*	*	*	*	*	−*	−*	*	(*)	*	*
G2.	*	*	*	*	*	*	*	−*	*	*	*	*
G3.	*	*	*	*	*	*	*	−*	*	*	*	*
G4.	*	*	*	*	(*)	*			*	(*)	(*)	*
G5.	*	*	*	*	*	*	*	−*	*	*	*	*
G6.	*		*	*			*	−*	*	*	*	*
G7.	*	*	*	*	*	*	*		*	*	*	*
G8.	*	*	*	*	*	*	*	−*	*	*	*	*
G9.	*	*	*	*	*	*	*	*	*	*	*	*
G10.	*		*	*	*	*			*	*	*	*
G11.	*			*	*	*	*	*	*	*	*	*
G12.	(*)	(*)	(*)	(*)	(*)	(*)			*	*	*	*
G13.			(*)	(*)	*	−*			*	(*)	(*)	*
G14.				*	(*)				*	(*)	(*)	*
G15.	(*)		*	*	(*)				*	(*)	(*)	*

Note: ∗ means that the protocol meets the stated goal, (∗) shows that the protocol can be modified to satisfy the requirement, 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.

The proposed protocol also meets the requirement for 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: they are signature keys. Therefore, even if eventually $\mathcal{A}$ finds out the signature key of any entity it will not enable him/her to determine past session keys. This independence of long term secrets from the session key generation process also enables the protocol to satisfy G7.

Communicating entities in the STCP share signed messages with each other that include the session information, thus providing mutual non-repudiation (G11). G14 is ensured by the inclusion in the protocol of the session cookie, which provides a limited protection against DoS, and by the fact that individual devices have pre-configurations of communication partners which enable them to drop a connection if an entity trying to connect with them is not able to authenticate.

To satisfy G15, the device identities are basically a random string that should not have any link with the function of the device. This would hinder an attacker from eavesdropping a protocol run to determine which SE is communicating on the wireless channel.

Finally, the TEMs of all SE-equipped UAVs provide trust validation proof signed by the TEM private key. This provides mutual validation of the trust between communicating devices, confirming that the other device is operating in a secure and reliable state (G13).

5.2 Revisiting the Requirements and Goals

Table 3 provides a comparison between the protocols listed in section 3.2 and the proposed protocol in terms of the required goals (see section 4.1).

As shown in Table 3, 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 attacks (G12) and privacy protection (G15). 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, G10, G11 and G13-G15.

The ASPeCT and JFK protocols meet a large set of goals. Both of these protocols can easily be modified to provide trust assurance (requiring additional signatures). Both of these protocols are vulnerable to partial chosen key attacks. However, in Table 3 we opt for the possibility that the ASPeCT and JFK protocols can be modified to meet this goal because in a fleet of UAVs all communicating SE may be of the same computation power and have a strong offline pre-deployment relationship.

The T2LS protocol meets the trust assurance goal by default. However, for the remaining goals it has the same results as the SSL protocol. A point in favour of the SCP81, MM, and SM protocols is that they were designed for the smart card industry where there is a strong and centralised organisational model. Most of these protocols, to some extent, have a similar architecture, in which a server generates the key and then communicates that key to the client. There is no non-repudiation as they do not use signatures in the protocol run.

Both SSH and SSL meet a large set of requirements and also have the potential to be extended to satisfy the additional requirements. However, to provide a flexible, backward compatible and universally acceptable architecture these protocols have too many optional parameters. Such flexibility is one of the main causes of most of the issues that these protocols have been plagued with in the last couple of years, heartbleed being the most infamous vulnerability.

The only difference between the P-STCP and the proposed protocol (except for the message structure) is the number of rounds to successfully complete a protocols run. P-SCTP has four messages (2-round protocol) and the proposed protocol uses 3 messages (1.5-round protocol).

As can be seen from Table 3, the proposed protocol satisfies all goals that were described in section 4.1.

5.3 Protocol Verification by CasperFDR and AVISPA

We selected the CasperFDR approach for formal analysis of the proposed protocol. The Casper compiler [28] takes as input a high-level description of the protocol, together with its security requirements along with the definition of an attacker and of its capabilities. The compiler 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 [30]. The intruder's capabilities modeled in the Casper script for the proposed protocol are as follows:

- an intruder can masquerade any entity in the network,
- an intruder can read the messages transmitted in the network, and
- an intruder cannot influence the internal process of an entity in the network.

The security specification for which CasperFDR evaluates the network is shown below. The listed specifications are defined in the #Specification section of the Casper script:

- the protocol run is fresh and both applications are alive,
- the key generated by the entity A is known only to the entity B (A and B are communication partners/devices),
- entities mutually authenticate each other and have mutual key assurance at the conclusion of the protocol,
- long-term keys of communicating entities are not compromised, and
- an intruder is unable to deduce the identities from observing the protocol messages.

The CasperFDR tool evaluated the protocol and did not find any feasible attack(s). The script is provided in appendix A.

Similarly, we scripted the proposed protocol in to High-Level Protocol Specification Language (HLPSL), as protocol description language for Automated Validation of Internet Security Protocols and Applications (AVISPA) [31]. The HLPSL is then translated into an intermediate language, which is an input to four different verifiers - as part of the AVISPA. These verifiers include SATMC, CL-AtSe, OFMC and TA4SP. Based on this analysis, no viable attack in the context of the operational environment of the protocol was found. The script is provided in appendix B.

6 Conclusion and Future Research Directions

In this paper, we outlined the concept of fleets of UAVs and discussed why such an architecture requires a secure channel for communication. The data communicated over a UAVs fleet has a strong requirement for confidentiality and integrity. To satisfy this requirement, communicating devices should have some cryptographic secrets to provide confidentiality and integrity. To generate these cryptographic secrets, the SEs run a secure channel protocol. In this paper, we proposed a secure channel protocol that not only provides mutual authentications and key sharing between the communicating entities but also provides assurance that each of the devices is in a secure and trusted state. We compared our proposed protocol with a list of selected protocols. Finally, we evaluated our protocol using CasperFDR and AVISPA, showing that it is secure against a number of attacks.

In future work, we will explore the major issues of detecting and neutralising wireless jamming and DoS attackers, along with building a strong mitigating framework.

Acknowledgments

The authors from XLIM acknowledge the support of:

– the SFD (Security of Fleets of Drones) project funded by Région Limousin;
– the TRUSTED (TRUSted TEstbed for Drones) project funded by the CNRS INS2I institute through the call 2016 PEPS ("Projet Exploratoire Premier Soutien") SISC ("Sécurité Informatique et des Systèmes Cyberphysiques");
– the SUITED (Suited secUrIty TEstbed for Drones), SUITED2 and UNITED (United NetworkIng TEstbed for Drones), UNITED2 projects funded by the MIRES (Mathématiques et leurs Interactions, Images et information numérique, Réseaux et Sécurité) CNRS research federation.

The authors from LaBRI acknowledge the support of:

– the TRUSTED (TRUSted TEstbed for Drones) project funded by the CNRS INS2I institute through the call 2016 PEPS ("Projet Exploratoire Premier Soutien") SISC ("Sécurité Informatique et des Systèmes Cyberphysiques");
– the SUITED-BX, SUITED2-BX and UNITED-BX, UNITED2-BX projects funded by LaBRI and its MUSe team.

References

1. R. N. Akram, P. F. Bonnefoi, S. Chaumette, K. Markantonakis, and D. Sauveron, "Secure autonomous uavs fleets by using new specific embedded secure elements," in *2016 IEEE Trustcom/BigDataSE/ISPA*, Aug 2016, pp. 606–614.
2. R. N. Akram, K. Markantonakis, and K. Mayes, "A paradigm shift in smart card ownership model," in *2010 International Conference on Computational Science and Its Applications*, March 2010, pp. 191–200.
3. ——, "Trusted platform module for smart cards," in *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*, March 2014, pp. 1–5.
4. "Smart Cards; Smart Card Platform Requirements Stage 1(Release 9)," European Telecommunications Standards Institute (ETSI), France, Technical Specification ETSI TS 102 412 (V9.1.0), June 2009. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/102400_102499/102412/09.01.00_60/ts_102412v090100p.pdf

5. 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.
6. “Trusted platform module main specification,” Trusted Computing Group, Tech. Rep., 2011.
7. L. Zhou and Z. Zhang, “Trusted Channels with Password-Based Authentication and TPM-Based Attestation,” *Communications and Mobile Computing, International Conference on*, vol. 1, pp. 223–227, 2010.
8. 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.
9. R. N. Akram, K. Markantonakis, and K. Mayes, “A Privacy Preserving Application Acquisition Protocol,” in *11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-12)*, G. Min and F. G. Marmol, Eds. Liverpool, United Kingdom: IEEE Computer Society, June 2012.
10. O. Blazy, P.-F. Bonnefoi, E. Conchon, D. Sauveron, R. N. Akram, K. Markantonakis, K. Mayes, and S. Chaumette, “An efficient protocol for uas security,” in *2017 Integrated Communications Navigation and Surveillance (ICNS)*, 2017.
11. J. A. Steinmann, R. F. Babiceanu, and R. Seker, “Uas security: Encryption key negotiation for partitioned data,” in *2016 Integrated Communications Navigation and Surveillance (ICNS)*, April 2016, pp. 1E4–1–1E4–7.
12. D. F. Pigatto, L. Gonçalves, G. F. Roberto, J. F. Rodrigues Filho, N. B. Floro da Silva, A. R. Pinto, and K. R. Lucas Jaquie Castelo Branco, “The hamster data communication architecture for unmanned aerial, ground and aquatic systems,” *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 705–723, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10846-016-0356-x
13. J. A. Maxa, M. S. B. Mahmoud, and N. Larrieu, “Extended verification of secure uaanet routing protocol,” in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sept 2016, pp. 1–16.
14. J. Won, S.-H. Seo, and E. Bertino, “A secure communication protocol for drones and smart objects,” in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS ’15. New York, NY, USA: ACM, 2015, pp. 249–260. [Online]. Available: http://doi.acm.org/10.1145/2714576.2714616
15. T. Dierks and E. Rescorla, “RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2,” Tech. Rep., August 2008.
16. W. Diffie, P. C. van Oorschot, and M. J. Wiener, “Authentication and Authenticated Key Exchanges,” *Designs, Codes and Cryptography*, vol. 2, no. 2, pp. 107–125, 1992.
17. A. Aziz and W. Diffie, “Privacy And Authentication For Wireless Local Area Networks,” *IEEE Personal Communications*, vol. 1, pp. 25–31, First Quarter 1994.
18. “Authentication and payment in future mobile systems,” ser. Lecture Notes in Computer Science, J.-J. Quisquater, Y. Deswarte, C. Meadows, and D. Gollmann, Eds. Springer Berlin / Heidelberg, 1998, vol. 1485, pp. 277–293, 10.1007/BFb0055870.
19. 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.

20. *Remote Application Management over HTTP, Card Specification v 2.2 - Amendment B*, Online, GlobalPlatform Specification, September 2006.
21. 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, September 2004, pp. 79–96.
22. 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.
23. 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-Verlag, 1997, pp. 30–45.
24. C. Mitchell, M. Ward, and P. Wilson, "Key Control in Key Agreement Protocols," *Electronics Letters*, vol. 34, no. 10, pp. 980 –981, May 1998.
25. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC, October 1996.
26. C. Furlani, *FIPS 186-3 : Digital Signature Standard (DSS)*, Online, National Institute of Standards and Technology (NIST) Std., June 2009. [Online]. Available: http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf
27. R. N. Akram, K. Markantonakis, and K. Mayes, *A Dynamic and Ubiquitous Smart Card Security Assurance and Validation Mechanism*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 161–172. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15257-3_15
28. G. Lowe, "Casper: a compiler for the analysis of security protocols," *J. Comput. Secur.*, vol. 6, pp. 53–84, January 1998.
29. C. A. R. Hoare, *Communicating sequential processes*. New York, NY, USA: ACM, 1978, vol. 21, no. 8.
30. P. Ryan and S. Schneider, *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley Professional, 2000.
31. A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P. H. Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron, "The avispa tool for the automated validation of internet security protocols and applications," in *Proceedings of the 17th International Conference on Computer Aided Verification*, ser. CAV'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 281–285. [Online]. Available: http://dx.doi.org/10.1007/11513988_27

Appendix A CasperFDR Script

```
#Free variables
datatype Field = Gen | Exp(Field, Num) unwinding 2
hkSE2, hkSE1, iMsg, rMsg, EnMaKey : Field
SE1, SE2, U: Agent
gSE1, gSE2: Num
nSE1, nSE2, SE1Val, SE2Val: Nonce
VKey: Agent->PublicKey
SKey: Agent->SecretKey
InverseKeys = (VKey, SKey), (EnMaKey, EnMaKey), (Gen, Gen), (Exp, Exp)
```

```
#Protocol description
0.    -> SE2 : SE1 [SE1!=SE2] <iMsg := Exp(Gen,gSE2)>
1. SE2 -> SE1 : SE2, nSE2, iMsg%hkSE2 <EnMaKey := Exp(hkSE2, gSE1); rMsg :=
Exp(Gen,gSE1)>
2. SE1 -> SE2 : nSE1, rMsg%hkSE1 <EnMaKey := Exp(hkSE1, gSE2)>
3. SE2 -> SE1 : nSE2, nSE1
4. SE1 -> SE2 : {{rMsg, U, nSE2}{SKey(U)}}{EnMaKey} [rMsg==hkSE2]
5. SE2 -> SE1 : {{iMsg,SE2, nSE1}{SKey(SE2)}}{EnMaKey} [iMsg==hkSE1]
6. SE1 -> SE2 : {{SE1OSHash, SE1, nSE2}{SKey(SE1)}}{EnMaKey}

#Actual variables
ADev1, ADev2, ME: Agent
GSE1, GSE2, GMalicious: Num
NSE1, NSE2, SE1VAL, SE2VAL, NMalicious: Nonce

#Processes
INITIATOR(SE2, SE1, U, SE2VAL, gSE2, nSE2) knows SKey(SE2), VKey
RESPONDER(SE1, SE2, U, SE1VAL, gSC, nSC) knows SKey(U), SKey(SC), VKey

#System
INITIATOR(ADev2, ADev1, ADev2Val, GSE2, NSE2)
RESPONDER(ADev1, ADev2, ADev1Val, GSE1, NSE1)

#Functions
symbolic VKey, SKey

#Intruder Information
Intruder = ME
IntruderKnowledge = {ADev2, ADev2, ME,
GMalicious, NMalicious, SKey(ME), VKey}

#Specification
Aliveness(SE2, SE1)
Aliveness(SE1, SE2)
Agreement(SE2, SE1, [EnMaKey])
Secret(SE2, EnMaKey, [SE1])
Secret(SE1, U, [SE2])

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

Appendix B AVISPA Script

```
role   se_1 (A,B: agent,
             G : text,
       PK1,PKTM1: public_key,
     CSE1,CTMSE1: message,
 H,Hk, MAC, SIGN: hash_func,
       SND, RCV : channel(dy))
played_by A
     def=
         local   NS1, NS2, Rs1, Rs2          : text,
                 State                       : nat,
                 VR1, VR2                    : text,
                 SAS1, SAS2, Sdata1, Sdata2  : text,
                 Success                     : text,
                 Svalid1, Svalid2            : text.text.text,
                 Kdh, Ke, Ka                 : message,
                 PK2, PKTM2                  : public_key,
                 Scookie                     : message,
                 CSE2, CTMSE2                : message
        const    sec_kdh1, sec_ke1, sec_ka1  : protocol_id
        init     State:= 0
transition
1.   State=0 /\ RCV(start) =|>
                 State':=2 /\ NS1':= new() /\ Rs1' := new()
                           /\  SND(A.B.NS1'.exp(G, Rs1').VR1.Scookie)
2.   State=2 /\ RCV(B.A.NS2'.exp(G, Rs2').Sdata2'.Svalid2'.
             {SIGN(Sdata2')}_(inv(PK2')).{SIGN(Svalid2')}_(inv(PKTM2')).
             MAC(Ka'.{{SIGN(Sdata2')}_(inv(PK2')).
             {SIGN(Svalid2')}_(inv(PKTM2')).CSE2'.CTMSE2'}_Ke')
             .VR2.Scookie) =|>
     State':= 4   /\  SAS1':= new()
                  /\ Svalid1':= SAS1'.NS2'.NS1
                  /\  Sdata1':= H(A.B.exp(G, Rs2').exp(G, Rs1).NS2'.NS1)
                  /\ Kdh' := exp(exp(G, Rs2'), Rs1)
                  /\ Ke':= {Hk(NS1.NS2'.1)}_Kdh'
                  /\ Ka':= {Hk(NS1.NS2'.2)}_Kdh'
                  /\ SND(A.B.Sdata1'.Svalid1'.{SIGN(Sdata1')}_(inv(PK1)).
                     {SIGN(Svalid1')}_(inv(PKTM1)).MAC(Ka'.
                     {{SIGN(Sdata1')}_(inv(PK1)).
                     {SIGN(Svalid1')}_(inv(PKTM1)).CSE1.CTMSE1}_Ke').Scookie)
                  /\ waitness ( A,B, ns1, NS1)
3. State= 4 /\ RCV(Success') =|>
              State':=6       /\ request (A,B, ns2, NS2)
                              /\ secret (Kdh,sec_kdh1, {A,B})
                              /\ secret(Ke, sec_ke1, {A,B})
                              /\ secret(Ka, sec_ka1, {A,B})
end role
role   se_2 (B,A: agent,
              G : text,
       PK2,PKTM2: public_key,
     CSE2,CTMSE2: message,
     H,Hk, MAC, SIGN: hash_func,
           SND, RCV : channel(dy))
played_by B
     def=
        local   NS2, NS1, Rs2, Rs1              : text,
        State                                   : nat,
          VR2, VR1                              : text,
          SAS2, SAS1, Sdata2, Sdata1            : text,
          Success                               : text,
          Svalid2, Svalid1                      : text.text.text,
          Kdh, Ke, Ka                           : message,
          PK1, PKTM1                            : public_key,
          Scookie                               : message,
          CSE1, CTMSE1                          : message
          const     sec_kdh2, sec_ke2, sec_ka2  : protocol_id
          init      State:= 1
transition
1.          State=1 /\ RCV(A.B.NS1'.exp(G, Rs1'). VR1.Scookie) =|>
            State':= 3 /\ NS2':= new()    /\ Rs2':=new()   /\  SAS2':= new()
                       /\ Sdata2':= H(B.A.exp(G, Rs1').exp(G, Rs2'). NS1'.NS2')
                       /\  Svalid2':= SAS2'.NS1'.NS2'
                       /\   Kdh' := exp(exp(G, Rs1'), Rs2')
                       /\ Ke' := {Hk(NS1.NS2'.1)}_Kdh'
                       /\ Ka' := {Hk(NS1.NS2'.2)}_Kdh'
                       /\ SND (B.A.NS2'.exp(G, Rs2'). Sdata2'.Svalid2'.
                       {SIGN(Sdata2')}_(inv(PK2)). {SIGN(Svalid2')}_(inv(PKTM2))
                       .MAC(Ka'.{{SIGN(Sdata2')}_(inv(PK2)).
                       {SIGN(Svalid2')}_(inv(PKTM2)).CSE2.CTMSE2}_Ke').VR2.Scookie)
                       /\ waitness (B,A, ns2, NS2')
2.        State=3 /\ RCV  (A.B.Sdata1'.Svalid1'.{SIGN(Sdata1')}_(inv(PK1')).
                  {SIGN(Svalid1')}_(inv(PKTM1')).MAC(Ka.{{SIGN(Sdata1')}_(inv(PK1')).
                  {SIGN(Svalid1')}_(inv(PKTM1')).CSE1'.CTMSE1'}_Ke).Scookie)
 =|>
             State':= 5 /\  request(B,A,ns1, NS1)
```

```
                        /\ SND(Success')
                        /\ secret (Kdh,sec_kdh2, {B,A})
                        /\ secret(Ke, sec_ke2, {B,A})
                        /\ secret(Ka, sec_ka2, {B,A})
end role
role session (A,B: agent,
                G: text,
           H,Hk, MAC, SIGN: hash_func)
def=
    local         SA, RA, SB, RB :     channel(dy),
       CSE1, CSE2, CTMSE1, CTMSE2: message,
           PK1, PK2, PKTM1, PKTM2:  public_key
    composition
    se_1(A,B, G, PK1, PKTM1, CSE1, CTMSE1, H,Hk, MAC, SIGN, SA, RA)
       /\    se_2(B, A, G, PK2, PKTM2, CSE2, CTMSE2, H, Hk, MAC, SIGN, SB,
RB)
end role
role environment( )  def=
         const   ns1, ns2                  : protocol_id,
                 a, b                      : agent,
                 pk1, pk2, pki             : public_key,
                 g                         : text,
                 h, hk, mac, sign          : hash_func
         intruder_knowledge = {a, b, i, pk1, pk2, pki, inv(pki), g, h, hk,
mac, sign}
         composition
                    session (a ,b, g, h, hk, mac, sign)
                /\  session (a ,i , g, h, hk, mac, sign)
                /\  session (i ,b , g,  h, hk, mac, sign)
end role
goal
secrecy_of    sec_kdh1,    sec_kdh2 , sec_ke1,    sec_ke2,    sec_ka1,    sec_ka2
authentication_on    ns1
authentication_on    ns2
end goal
environment ()
```