An Efficient, Secure and Trusted Channel Protocol for Avionics Wireless Networks

Raja Naeem Akram[†], Konstantinos Markantonakis[†], Keith Mayes[†]
Pierre-François Bonnefoi[‡], Damien Sauveron[‡§] and Serge Chaumette[§]
[†]*Information Security Group Smart Card Centre, Royal Holloway, University of London, Egham, United Kingdom*
[‡]*XLIM (UMR CNRS 7252 / Université de Limoges), Département Mathématiques Informatique. Limoges, France*
[§]*LaBRI (UMR CNRS 5800 / Université de Bordeaux), Talence, France*
Email: {r.n.akram, k.markantonakis, keith.mayes}@rhul.ac.uk,
{pierre-francois.bonnefoi, damien.sauveron}@unilim.fr, serge.chaumette@labri.fr

***Abstract*—Avionics networks rely on a set of stringent reliability and safety requirements. In existing deployments, most of these networks are based on a wired technology, which supports these requirements. Furthermore, this technology simplifies the security management of the network since certain assumptions can be safely made, including the inability of an attacker to access the network, and the fact that it is almost impossible for an attacker to introduce a node into the network. The proposal for Avionics Wireless Networks (AWNs, currently under consideration by multiple aerospace working groups, promises a reduction in the complexity of electrical wiring harness design and fabrication, a reduction in the total weight of wires, increased customization possibilities, and the capacity to monitor otherwise inaccessible moving or rotating aircraft parts such as landing gear and some sections of the aircraft engines. While providing these benefits, the AWN must ensure that it provides levels of safety that are at minimum equivalent to those offered by the wired equivalent. In this paper, we propose a secure and trusted channel protocol that satisfies the stated security and operational requirements for an AWN protocol. There are three main objectives for this protocol. First, the protocol has to provide the assurance that all communicating entities can trust each other, and can trust their internal (secure) software and hardware states. Second, the protocol has to establish a fair key exchange between all communicating entities so as to provide a secure channel. Finally, the third objective is to be efficient for both the initial start-up of the network and when resuming a session after a cold and/or warm restart of a node. The proposed protocol is implemented within a demo AWN, and performance measurements are presented based on this implementation. In addition, we formally verify our proposed protocol using CasperFDR.**

1. Introduction

A modern aircraft can be considered as a highly reliable and mission-critical digital network in the air. The Aircraft Data Network (ADN) interconnects different aircraft sub-systems, including flight control, the crew network and the passenger entertainment network. In recent years investigations into the feasibility of moving some non-critical networks from wired technology to wireless-based technology have been carried out. Such a network is referred to as an Avionics Wireless Network (AWN), which is the main focus of this paper.

Whatever the network deployment topology and the communication technology that are used, one element is common: the physical wire that connects two or more avionics sub-systems. Wiring an aircraft can be costly in that it includes wiring harness designs, cable fabrication and the associated cost of additional weight. Furthermore, to provide dual redundancy, these wires have to connect any two devices by means of two physically separate paths in the aircraft. Wires and related connectors potentially represent 2-5 percent of an aircraft's weight [1]. As the wiring of an aircraft is a time- and labor-intensive activity, post-deployment upgrades or installation of new wire routes or new avionics sub-systems may be costly [2]. As reported by [1], roughly 30 percent of wires are potential candidates for wireless substitutes. Therefore, as highlighted in [3], wireless solutions have more than reasonable prospects as long as security, safety and high reliability can be maintained.

Whether an ADN or an AWN is used, the main objective is to communicate data between aircraft sub-systems in a secure, reliable and efficient manner. Going wireless brings its own set of unique challenges, among which a major one is to ensure the confidentiality and integrity of communications; any attacker within wireless range of the AWN can easily eavesdrop and/or (potentially) modify the exchanged information. To protect against such an attack, we require a strong, efficient and trustworthy mechanism to establish secure links between the communicating nodes in an AWN. Secure channel protocols can be used for this purpose, and in this paper we propose such a protocol for AWN environments. In this paper, we are not going to discuss the wireless jamming attacks. Although they are a valid threat but they do not directly attack the confidentiality and integrity of communication channel - wireless jamming attack is a thread to channel availability. For this reason they are beyond the scope of this paper.

1.1. Contribution

In this paper, our main goals are to propose a secure and trusted channel protocol for AWNs, and to compare its security and performance with several other existing protocols.

The salient contributions of this paper are as follows:

1) proposing a Secure and Trusted Channel Protocol (STCP) that along with establishing a secure channel between the communicating entities (end-points) also provides security assurance that each end-point is secure and trusted;
2) defining comparison criteria for secure channel protocols along with the related security and performance analysis;
3) validating the proposed protocol with a formal tool, CasperFDR and producing an implementation in a real AWN to enable measurements to be obtained.

1.2. Structure of the Paper

Section 2 briefly presents the rationale behind this paper and the existing work carried out in the avionics industry (in the context of AWNs) and secure channel protocols from a traditional computer security perspective. In section 3, we look into how a Trusted Platform Module (TPM) can provide a trusted boot that is then used to assure communication partners that the device is secure and trustworthy. Section 4 discussed the security comparison criteria and then the proposed protocol. In section 5, we first analyze the proposed protocol informally, than formally using CasperFDR and we compare it with different protocols based on the security comparison criteria previously defined. Finally in section 6 we present future research directions and conclude the paper.

2. Rationale and Related Work

In this section, we discuss the rationale behind the proposed protocol and review the existing work in two different areas: AWNs and Secure Channel Protocols (SCPs).

2.1. Rationale

A Secure Channel Protocol (SCP) by definition provides either or both of entity authentication and key exchange between communicating parties (end points). An 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 avionics it is not just implicit trust that should be required but also explicit trust validation, to counter any potential threat. The explicit trust assurance should be provided by the (aircraft) device that is participating in the AWN communication. This would build in an assurance that only secure and trusted devices (explicitly trusted devices with per-protocol run assurance) will participate in the AWN, potentially countering physically altered devices and/or re-introduction of a decommissioned device as discussed in [3, 5].

In contrast, in the ADN, the assumption of implicit assurance might be valid. However, for a robust security and reliability mechanism an explicit security assurance mechanism should be considered.

A trusted channel is a secure channel that is cryptographically bounded to the current state of the communicating parties [6]. This state can be a hardware and/or a software configuration, and ideally it requires a trustworthy component to validate it is effectively as claimed. Such a component, in most instances, is a TPM [7] as demonstrated in [8]–[10]

In an AWN, individual devices will have prior relationships with each other: in the avionics industry any system deployment is stringently controlled, regulated and protected. Therefore, assuming that one single trusted entity would deploy the AWN environment is as per the avionics industry's practice. However, 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.

2.2. Related Work on AWN Security Concerns

Security and trust have been subject to some analysis by both the academic community and the industry. A brief overview of aircraft information security and some improvements were proposed in [11]. Security assurance research from airplane production to airplane operation was presented in [12, 13]. A general discussion of the security issues related to the aircraft network and aircrafts' connectivity with the Internet is provided in [14], while [15, 16] discusses the impact of WSNs (Wireless Sensor Networks) and related security concerns in aircraft. Security and safety are intrinsically linked to each other in general and specifically in the context of the aviation industry [17]–[19]. The application and impact of cryptography, especially public key cryptography for avionics networks, was evaluated in [20].

The management of security and the general deployment of AWNs based on wireless-as-a-comm-link have been analyzed in [3], which discusses the security and trust challenges faced by AWNs. In addition, a crucial component that supports aircraft devices security is the trusted boot process discussed in [5]. The security, trust and assurance issues related to the fact of bringing a user device into an aircraft network are evaluated in [21].

2.3. 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 points of comparison in the discussions to come.

The concept of trusted channel protocols was proposed by Gasmi et al. [6] along with the adaptation of the TLS protocol [22]. Later Armknecht et al. [9] proposed another adaptation of OpenSSL to accommodate the concept of trusted channels; similarly, Zhou and Zhang [8] also proposed an 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 [23], the Aziz-Diffie (AD) protocol [24], the ASPeCT protocol [25], Just-Fast-Keying (JFK) [26], trusted TLS (T2LS) [6], GlobalPlatform SCP81 [27], the Markantonakis-Mayes (MM) protocol [28], and the Sirett-Mayes (SM) protocol [29].

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 mobile networks' value-added services. 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 embedded low-power devices. In addition, we have included the secure and trusted channel protocol, P-STCP [10], which is designed for resource-restricted and security-sensitive environments, and has some similar design requirements to those of the proposed protocol.

3. Trusting a Device (Trusted Boot)

In this section, we discuss how a TPM provides a secure boot process and how it provides assurance to external entities that the device is secure and trustworthy.

3.1. Trusted Platform Module

The TPM is a trusted, reliable and tamper-resistant component that can provide trustworthy evidence of the state of a given system on which it is present. The interpretation of this evidence is neither controlled nor dictated by the TPM but by the entity receiving and thus assessing it. Trust in this context can be defined as an expectation that the state of a system is as it is supposed to be, i.e. secure. Therefore, in a very simplistic sense a TPM is a trustworthy reporting agent (witness), not an evaluator or an enforcer of security policies. In the field of trusted computing, this is referred to as providing a root of trust on which an inquisitor relies to validate the current state of a system.

For in-depth discussion of the architecture of TPMs and their functionality please refer to [7]. In this paper, we focus on the secure boot process as it is carried out by the TPM and as discussed in the subsequent section.

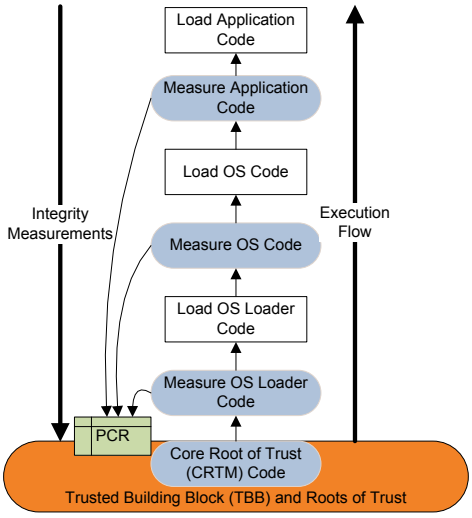

Figure 1. Trusted Platform Boot Sequence (figure from [30])

3.2. Secure Boot (TPM Integrity Measurement Operation)

When a device with a TPM boots up, the first component to power up is the system BIOS (Basic Input/output System). On a trusted platform (a platform that contains a TPM), the boot sequence is initiated by the Core BIOS (*i.e.* CRTM: Core Root of Trust Measurement), which first measures its own integrity. This measurement is stored in PCR_0[1] and it is later extended to include the integrity measurement of the rest of the BIOS. The Core BIOS then measures the circuit-board's (motherboard) configuration setting[2], and this value is stored in PCR_1. After these measurements, the Core BIOS loads the rest of the code of the BIOS.

The BIOS will subsequently measure the integrity of the ROM firmware and of the ROM firmware configuration, storing them in PCR_2 and PCR_3 respectively. At this stage, the base configuration of a device is established and the CRTM will proceed with integrity measurement and loading of the Operating System (OS).

The CRTM measures the integrity of the "OS Loader Code", also termed the Initial Program Loader (IPL), and

1. A Platform Configuration Register (PCR) is a 160-bit (20 bytes) data element that stores the result of the integrity measurement, which is a generated hash of a given component (*e.g.* the BIOS, the operating system, or an application). A group of PCRs form the integrity matrix. The process of extending PCR values is as follows: $PCR_i = Hash(PCR'_i||X)$, where i is the PCR index, PCR'_i represents the old value stored at index i, and X is the sequence to be included in the PCR value. "||" indicates the concatenation of two data elements in the given order. The starting value of all PCRs is zero.

2. To measure that correct hardware configuration was present at boot time.

stores the measurement in the relevant PCR. The designated PCR index is left to the discretion of the OS developers. Subsequently, the device will execute the "OS Loader Code" and if successful, the TPM will measure the integrity of the "OS Code". After this measurement is made and stored, the "OS Code" executes. Finally, the relevant software that initiates its execution will first be subjected to an integrity measurement, and the resulting value will be stored in a PCR and then the software will be allowed to execute. This process is shown in Figure 1, which illustrates the execution flow and the storage of the integrity measurements.

By creating a chain of integrity measurements, a TPM provides a trusted and reliable view of the current state of the system. Any piece of software, whether part of the OS or an application, has an integrity measurement stored in a PCR at a particular index.

As discussed above, a TPM does not make any decisions: it only measures, stores, and reports integrity measurements in a secure and reliable manner. When a TPM reports an integrity measurement, it is recommended that it generates a signature on the value, thus avoiding replay and man-in-the-middle attacks [7]. The process by which an inquisitor can request a device attestation and how a TPM provides this evidence is discussed in the next section.

Reporting and Attestation Operations. The attestation process, whether initiated by the relevant external entity (including human users or other devices) locally or remotely, involves the generation of a signature by the TPM using the Attestation Identification Key (AIK) of the (associated/requested) PCR values [30]. The signature assures the requesting entity of the validity of the integrity measurement stored in the PCRs. The choice of the AIK and PCR index is dependent on the device, OS or application developer.

The signature key and PCR values are stored in a tamper-resistant memory inside the TPM. Therefore, an attacker would have to circumvent the tamper-resistant property of the TPM to impact the outcome of this attestation process.

4. Secure and Trusted Channel Protocol

In this section, we begin the discussion with the security comparison criteria, followed by the protocol notation, 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 AWN framework, 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.

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 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 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 $\mathcal{Z} \neq \mathcal{X}$. 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 to impersonate $\mathcal{Y}$. Nevertheless, key compromise should not enable him to impersonate other entities to $\mathcal{Y}$ [31].

G10) ***Perfect Forward Secrecy*:** If the long-term keys of 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 [32]. 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, 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 AWN nodes.

For a formal definition of the terms (italicized) used in the above list, the reader is referred to [33]. The requirements listed above are later used as a point of reference to compare the selected protocols in Table 3.

For the performance evaluation that we have conducted, the main measurements are related to the time required to establish a secure channel once the wireless link is established and they are discussed in section 5.3.

4.2. Protocol Notation

The notations used in the protocol description are listed in Table 2;

4.3. Pre-Protocol Setup

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

1) Each aircraft device that is part of the AWN has a TPM.
2) Each device in the AWN is pre-configured with the signature verification keys of its communication partners (*i.e.* public keys of other aircraft devices).
3) Each device is also pre-configured with the signature verification keys of the TPMs of its communication partners (*i.e.* the public key corresponding to the AIK key used to sign the PCR values stored in the TPM) along with their own trusted and secure PCR values (*i.e.* the values for their trusted and secure state).

4.4. Proposed Protocol

The messages of the protocol are listed in Table 1 and described below.

Message 1. The AD1 generates a random number N_{AD1} and computes the Diffie-Hellman exponential $g^{r_{AD1}}$. The "$H(g^{r_{AD1}} \| N_{AD1} \| AD1_i \| AD2_i)$" 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 (possibly) provides the protocol session resumption facility, which is required if a protocol run is interrupted before it successfully concludes. Finally, AD1 will request AD2 to provide assurance of its current state.

Message 2. In response, AD2 generates a random number, and a Diffie-Hellman exponential $g^{r_{AD2}}$. It can then calculate the $k_{DH} = (g^{r_{AD1}})^{r_{AD2}} \ (mod\ n)$ which will be the shared secret from which the rest of the keys will be generated. The encryption key is generated as $K_e = H_{k_{DH}}(N_{AD1} \| N_{AD2} \| ''1'')$ and a MAC key as $K_a = H_{k_{DH}}(N_{AD1} \| N_{AD2} \| ''2'')$. We can further generate (session) keys in a similar manner for data stream-specific virtual links[3] (VLs) for managing the communication between different aircraft sub-systems.

3. Virtual Links (VLs): Each communication relationship in an aircraft network is represented as a VL. In our proposal we assume that a pair of communication parties would have two uni-directional VLs and each VL will have its own session key.

Subsequently, the TPM generates a state validation message signed by the TPM AIK key represented in the protocol as "$Sign_{TPM_{AD2}}(AD2 - Validation)$". AD2 will also request AD1 to provide assurance of its current state.

On receipt of this message, AD1 will first generate the session keys. AD1 will then verify AD2's signature and validation proof generated by the TPM of AD2. As the signature key belongs to the TPM of AD2, an attacker cannot masquerade this signature. By verifying the signature, AD1 can ascertain the current state (PCR value) is measured by the TPM of AD2. Now AD1 can verify whether the PCR value represents a trusted and secure state or not. Since our protocol pre-setup AD1 would have the PCR value of a trusted and secure state of AD2.

Furthermore, AD1 will check the values of Diffie-Hellman exponentials (*i.e.* $g^{r_{AD1}}$ and $g^{r_{AD2}}$) and of the generated random numbers to avoid main-in-the-middle and replay attacks.

Message 3. AD1 will then generate a message similar to message 2, a signature by AD1 and trust validation proof generated by its TPM.

On receipt of the message, AD2 will verify the trust validation proof and generate keys. Furthermore, AD2 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 session encryption and MAC keys can be saved as master session keys. Individual VL keys can then be generated from these session keys. Based on the security policies related to the VLs, whether they require only confidentiality or integrity or both, these two master session keys can be used to generate VL specific encryption and MAC keys.

4.6. Protocol Resumption

As discussed in [3], secure channel protocols only run when an aircraft is stationary on the ground, with proofs that the aircraft is not in flight based on geo-location, proximity to airport, weight on wheels, etc. The proposed protocol would run before each flight and master session keys are only valid for a single flight. The protocol should not be executed during the flight. Therefore, if a device has to reset due to some unforeseeable situation, a safety procedure to resume the secure channel and all of the associated VL keys - without running the protocol - must exist. For this purpose, each individual device will save the master session keys in its persistent storage and will have a standard algorithm to generate the keys for each of the VLs. If the master session keys are lost, then, during that particular flight, the device would be out of communication. To avoid this, the master session keys should be stored on two different memories

Table 1. SECURE AND TRUSTED CHANNEL PROTOCOL (STCP).

1.	$AD1 \rightarrow AD2$	:	$AD1_i \| AD2_i \| N_{AD1} \| g^{r_{AD1}} \| VR_{AD1-AD2} \| S_{Cookie}$
2.	$AD2 \rightarrow AD1$	:	$AD2_i \| AD1_i \| N_{AD2} \| g^{r_{AD2}} \| [Sign_{AD2}(AD2-Data) \| Sign_{TPM_{AD2}}(AD2-Validation)]_{K_a}^{K_e} \| VR_{AD2-AD1} \| S_{Cookie}$
		:	$AD2-Data = H(AD2_i \| AD1_i \| g^{r_{AD1}} \| g^{r_{AD2}} \| N_{AD1} \| N_{AD2})$
		:	$AD2-Validation = SAS_{AD2-AD1} \| N_{AD1} \| N_{AD2}$
3.	$AD1 \rightarrow AD2$	:	$[Sign_{AD1}(AD1-Data) \| Sign_{TPM_{AD1}}(AD1-Validation)]_{K_a}^{K_e} \| S_{Cookie}$
		:	$AD1-Data = H(AD1_i \| AD2_i \| g^{r_{AD2}} \| g^{r_{AD1}} \| N_{AD2} \| N_{AD1})$
		:	$AD1-Validation = SAS_{AD1-AD2} \| N_{AD2} \| N_{AD1}$

Table 2. NOTATION USED IN PROTOCOL DESCRIPTION.

$AD1$	:	Denotes an aircraft device '1'.
$AD2$	:	Denotes an aircraft device '2'.
$A \rightarrow B$	:	Message sent by an entity A to an entity B.
TPM_X	:	Denotes a TPM of an entity X
X_i	:	Represents the identity of an entity X.
g^{rx}	:	Diffie-Hellman exponential generated by 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 [34].
$H(Z)$	:	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 (PCR values) generation by entity A that provides trust validation to the requesting entity B.

(each aircraft device has at least two separate storage media, so as to provide this dual storage redundancy).

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 some implementation results and a formal analysis using CasperFDR.

5.1. Brief Information 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 TPM) before the aircraft devices are deployed (pre-deployment configuration).

The proposed protocol satisfies requirements G3, G4, G5 and G12 by first requiring AD1 and AD2 to generate the Diffie-Hellman exponential; thus computational cost is equal on both sides. Similarly, exponential generation also assures that both devices will have equal input to the key generation. Messages 2 and 3 are encrypted used the keys generated during the protocol, 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 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 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.

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 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 mes-

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	Asymmetric TKDF	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.

sages 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 aircraft device is communicating on the wireless channel.

Finally, TPMs on all communicating devices provide trust validation proof in the form of PCR values signed by the TPM AIK. This provides mutual validation of the trust between communicating devices, confirming that the other device is functioning in a secure and reliable state (G13).

5.2. Revisiting the Requirements and Goals

Table 3 provides a comparison between the listed protocols in section 2.3 with 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 and G13-G15.

The ASPeCT and JFK protocols meet a large set of goals. Both of these protocols can be easily 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 an AWN all communicating devices 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 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.

Asymmetric TKDF (Trusted Key Distribution Frameworks) does not satisfy a number of requirements. In contrast, P-STCP satisfies most of the requirements listed in the table. 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 apparent from the table 3, the proposed protocol satisfies all goals that were described in section 4.1.

5.3. Practical Implementation

In our AWN test-bed each node is a Raspberry Pi model B supplied with a Wi-Fi USB dongle TL-WN722N by TP-

LINK. In all the measurements we made, the nodes were configured in ad-hoc mode.

For all the selected protocols, in our evaluation implementations, we setup two neighboring nodes to establish a secure channel. This provides a performance measurement of the protocols between individual communicating pairs. However for the TKDF, a key distribution server is also required and a third node in the ad-hoc network plays this role.

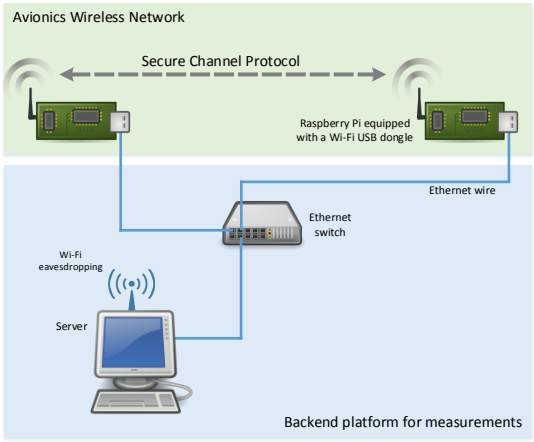

Figure 2. AWN test-bed

In our AWN test-bed, each node is connected to a backend server by means of an Ethernet connection. This server controls the nodes so as to prepare them for the target scenario and is also in charge of collecting the measurements. Effective measurement can be done internally on the node initiating the secure channel, called a client, and/or it can be done at the level of the network data exchanged between the nodes of the AWN and captured with a Wi-Fi card on the backend server set in monitor mode.

The performance comparison is provided in Table 4, comparing a subset of protocols from table 3 and proposed protocol performance in the developed test-bed environment.

Table 4. PROTOCOL PERFORMANCE MEASURES (MILLISECONDS)

SSL	SSH	Asymmetric TKDF	Proposed Protocol
1310.93	911.21	14447.63	4582.44

Note: Above-mentioned measurement values for SSL, SSH and Asymmetric TKDF are from [38].

In our Python implementation of the proposed protocol, the TPM was emulated by the Raspberry Pi. Key sizes used for our proposed protocol were 2048 bits MODP group for the Diffie-Hellman key generation, 2048 bits for RSA and 256 bits for symmetric encryption and MAC computation (AES).

The P-STCP protocol was implemented with smaller key sizes in [10], resulting in 2998.71ms performance measurement. Use the key sizes from [10] in our implementation results the performance of the proposed protocol to be 1201.50ms.

5.4. Protocol Verification by CasperFDR

We selected the CasperFDR approach for formal analysis of the proposed protocol. The Casper compiler [35] takes input as a high-level description of the protocol, together with its security requirements along with the definition of an attacker and its capabilities. The compiler then translates the description into the process algebra of Communicating Sequential Processes (CSP) [36]. The CSP description of the protocol can be machine-verified using the Failures-Divergence Refinement (FDR) model checker [37]. The intruder's capability modelled in the Casper script (appendix A) for the proposed protocol is:

- 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 as shown below. The listed specifications are defined in the #Specification section of appendix A:

- 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.

6. Conclusion and Future Research Directions

In this paper, we outlined the concept of the AWN and discussed why such a proposal requires a secure channel for communication. The data communicated over an AWN 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 devices 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 and experimental performance results were provided. Finally, we evaluated the protocol using CasperFDR, showing that our protocol is secure against a number of attacks.

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

addition to the trusted boot, for robust and reliable security we need to look into secure execution on AWN nodes - especially investigating the inclusion of ARM TrustZone and Intel SGX technologies.

7. Acknowledgments

The authors from Royal Holloway University of London acknowledge the support of the UK's innovation agency, InnovateUK, and the contributions of the SHAWN project partners. 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 Cyber-physiques");
- the SUITED (Suited secUrIty TEstbed for Drones) and UNITED (United NetworkIng TEstbed for Drones) 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 Cyber-physiques");
- the SUITED-BX and UNITED-BX projects funded by LaBRI and its MUSe team.

Disclaimer

The views and opinions expressed in this article are those of the authors and do not necessarily reflect the position of SHAWN project or any of organisations associated with this project.

References

[1] "Technical characteristics and operational objectives for wireless avionics intra-communications (waic)," ITU- R: Radiocommunication Sector of ITU, Tech. Rep. ITU-R M.2197, November 2010. [Online]. Available: http://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2197-2010-PDF-E.pdf

[2] D.-K. Dang, A. Mifdaoui, and T. Gayraud, "Fly-by-wireless for next generation aircraft: Challenges and potential solutions," in *Wireless Days (WD), 2012 IFIP*, Nov 2012, pp. 1–8.

[3] R. N. Akram, K. Markantonakis, S. Kariyawasam, S. Ayub, A. Seeam, and R. Atkinson, "Challenges of security and trust in avionics wireless networks," in *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, Sept 2015, pp. 4B1–1–4B1–12.

[4] "Smart Cards; Smart Card Platform Requirements Stage 1(Release 9)," ETSI, France, Tech. Rep. ETSI TS 102 412 (V9.1.0), June 2009.

[5] K. Markantonakis and R. N. Akram, "A secure and trusted boot process for avionics wireless networks," in *2016 Integrated Communications Navigation and Surveillance (ICNS)*, April 2016, pp. 1C3–1–1C3–9.

[6] 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.

[7] "Trusted platform module main specification," Trusted Computing Group, Tech. Rep., 2011.

[8] 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.

[9] 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.

[10] 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)*, F. G. M. Geyong Min, Ed. Liverpool, United Kingdom: IEEE Computer Society, June 2012.

[11] M. Olive, R. Oishi, and S. Arentz, "Commercial aircraft information security-an overview of arinc report 811," in *25th Digital Avionics Systems Conference, 2006 IEEE/AIAA*, Oct 2006, pp. 1–12.

[12] S. Lintelman, R. Robinson, M. Li, D. v. Oheimb, R. Poovendran, and K. Sampigethaya, "Security assurance for it infrastructure supporting airplane production, maintenance, and operation," in *Proc. U.S. National Workshop on Aviation Software Systems: Design for Certifiably Dependable Systems (NITRD HCSS-AS), 4-5 Oct 2006, Alexandria, VA*, J. Sprinkle, Ed., 2006, available from http://ddvo.net/papers/HCSS-AS.html.

[13] G. Ladstaetter, N. Reichert, and T. Obert, "It security management of aircraft in operation: A manufacturer's view," SAE Technical Paper, Tech. Rep., 2011.

[14] N. Thanthry, M. S. Ali, and R. Pendse, "Security, internet connectivity and aircraft data networks," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 21, no. 11, pp. 3–7, 2006.

[15] N. Thanthry and R. Pendse, "Aviation data networks: security issues and network architecture," in *Security Technology, 2004. 38th Annual 2004 International Carnahan Conference on*, Oct 2004, pp. 77–81.

[16] R. V. Robinson, K. Sampigethaya, M. Li, S. Lintelman, R. Poovendran, and D. von Oheimb, "Secure network-enabled commercial airplane operations: It support infrastructure challenges," in *Proceedings of the First CEAS European Air and Space Conference Century Perspectives (CEAS)*, 2007.

[17] S. Brostoff and M. A. Sasse, "Safe and sound: a safety-critical approach to security," in *Proceedings of the 2001 workshop on New security paradigms*. ACM, 2001, pp. 41–50.

[18] A. Pfitzmann, "Why safety and security should and will merge." in *SAFECOMP*, ser. Lecture Notes in Computer Science, M. Heisel, P. Liggesmeyer, and S. Wittmann, Eds., vol. 3219. Springer, 2004, pp. 1–2. [Online]. Available: http://dblp.uni-trier.de/db/conf/safecomp/safecomp2004.html#Pfitzmann04

[19] M. Paulitsch, R. Reiger, L. Strigini, and R. E. Bloomfield, "Evidence-based security in aerospace: From safety to security and back again." in *ISSRE Workshops*. IEEE, 2012, pp. 21–22. [Online]. Available: http://dblp.uni-trier.de/db/conf/issre/issre2012w.html#PaulitschRSB12

[20] R. Robinson, M. Li, S. Lintelman, K. Sampigethaya, R. Poovendran, D. von Oheimb, and J.-U. Bußer, "Impact of public key enabled applications on the operation and maintenance of commercial airplanes," in *Proc. of the 7th AIAA Aviation Technology, Integration and Operations Conference (ATIO)*. AIAA, 2007, http://ddvo.net/papers/AIAA_ATIO.html.

[21] R. N. Akram and K. Markantonakis, "Challenges of security and trust of mobile devices as digital avionics component," in *2016 Integrated Communications Navigation and Surveillance (ICNS)*, April 2016, pp. 1C4–1–1C4–11.

[22] T. Dierks and E. Rescorla, "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," Tech. Rep., August 2008.

[23] 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.

[24] A. Aziz and W. Diffie, "Privacy And Authentication For Wireless Local Area Networks," *IEEE Personal Communications*, vol. 1, pp. 25–31, First Quarter 1994.

[25] G. Horn and B. Preneel, "Authentication and payment in future mobile systems," in *Computer Security - ESORICS 98*, 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.

[26] 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.

[27] *Remote Application Management over HTTP*, Online, GlobalPlatform Specification, September 2006.

[28] 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.

[29] 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.

[30] R. N. Akram, K. Markantonakis, and K. Mayes, "An Introduction to the Trusted Platform Module and Mobile Trusted Module," in *Secure Smart Embedded Devices, Platforms and Applications*. Springer New York, 2014, pp. 71–93.

[31] 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. [Online]. Available: http://portal.acm.org/citation.cfm?id=647993.742138

[32] C. Mitchell, M. Ward, and P. Wilson, "Key Control in Key Agreement Protocols," *Electronics Letters*, vol. 34, no. 10, pp. 980 –981, May 1998.

[33] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC, October 1996.

[34] *FIPS 186-3 : Digital Signature Standard (DSS)*, Online, National Institute of Standards and Technology (NIST) Std., June 2009.

[35] G. Lowe, "Casper: a compiler for the analysis of security protocols," *J. Comput. Secur.*, vol. 6, pp. 53–84, January 1998. [Online]. Available: http://dl.acm.org/citation.cfm?id=353677.353680

[36] C. A. R. Hoare, *Communicating sequential processes*. New York, NY, USA: ACM, 1978, vol. 21, no. 8.

[37] P. Ryan and S. Schneider, *The Modelling and Analysis of Security Protocols: the CSP Approach*. Addison-Wesley Professional, 2000.

[38] R. N. Akram, K. Markantonakis, K. Mayes, P-F. Bonnefoi, D. Sauveron, and S. Chaumette, "Security and performance comparison of different secure channel protocols for Avionics Wireless Networks," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, Sept 2016.

Appendix A.
CasperFDR Script

```
#Free variables
datatype Field = Gen | Exp(Field, Num)
unwinding 2
hkAD2, hkAD1, iMsg, rMsg, EnMaKey : Field
AD1, AD2, U: Agent
gAD1, gAD2: Num
nAD1, nAD2, AD1Val, AD2Val: Nonce
VKey: Agent->PublicKey
SKey: Agent->SecretKey
InverseKeys = (VKey, SKey), (EnMaKey,
EnMaKey), (Gen, Gen), (Exp, Exp)

#Protocol description
0.    -> AD2 : AD1 [AD1!=AD2] <iMsg :=
Exp(Gen,gAD2)>
1. AD2 -> AD1 : AD2, nAD2, iMsg%hkAD2
<EnMaKey := Exp(hkAD2, gAD1); rMsg :=
Exp(Gen,gAD1)>
2. AD1 -> AD2 : nAD1, rMsg%hkAD1 <EnMaKey :=
Exp(hkAD1, gAD2)>
3. AD2 -> AD1: nAD2, nAD1
4. AD1 -> AD2 : {{rMsg, U,
nAD2}{SKey(U)}}{EnMaKey} [rMsg==hkAD2]
5. AD2 -> AD1 : {{iMsg,AD2,
nAD1}{SKey(AD2)}}{EnMaKey} [iMsg==hkAD1]
6.AD1 -> AD2 : {{AD1OSHash, AD1,
nAD2}{SKey(AD1)}}{EnMaKey}

#Actual variables
ADev1, ADev2, ME: Agent
GAD1, GAD2, GMalicious: Num
NAD1, NAD2, AD1VAL, AD2VAL, NMalicious: Nonce

#Processes
INITIATOR(AD2,AD1, U, AD2VAL, gAD2,
nAD2)knows SKey(AD2), VKey
READ2ONDER(AD1,AD2, U, AD1VAL, gSC, nSC)
knows SKey(U), SKey(SC), VKey

#System
INITIATOR(ADev2, ADev1,ADev2Val, GAD2, NAD2)
READ2ONDER(ADev1, ADev2, ADev1Val, GAD1,
NAD1)

#Functions
symbolic VKey, SKey

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

#Specification
Aliveness(AD2, AD1)
Aliveness(AD1, AD2)
Agreement(AD2, AD1, [EnMaKey])
Secret(AD2, EnMaKey, [AD1])
Secret(AD1, U, [AD2])

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