Post-Quantum Ready Key Agreement for Aviation

Marcel Tiepelt[1], Christian Martin[1] and Nils Maeurer[2]

[1] Karlsruhe Institute of Technology, KASTEL, Karlsruhe, Germany
[2] Airbus, Airbus Defence and Space, Taufkirchen, Germany

Abstract. Transitioning from classically to quantum secure key agreement protocols may require to exchange fundamental components, for example, exchanging Diffie-Hellman-like key exchange with a key encapsulation mechanism (KEM). Accordingly, the corresponding security proof can no longer rely on the Diffie-Hellman assumption, thus invalidating some security guarantees. As a consequence, the security properties have to be re-proven under a corresponding security notion.
We initiate the study of the LDACS key agreement protocol (Edition 01.01.00 from 25.04.2023), which is soon-to-be-standardized by the International Civil Aviation Organization. The protocol's cipher suite features Diffie-Hellman as well as a KEM-based key agreement protocol to provide post-quantum security. While the former results in an instantiation of an ISO key agreement inheriting all security properties, the security achieved by the latter is ambiguous.
We formalize the computational security using the systematic notions of de Saint Guilhem, Fischlin and Warinshi (CSF '20), and prove the exact security that the KEM-based variant achieves in this model; primarily entity authentication, key secrecy and key authentication. To further strengthen our "pen-and-paper" findings, we model the protocol and its security guarantees using Tamarin, providing an automated proof of the security against a Dolev-Yao attacker.

Keywords: authenticated key exchange · post-quantum key exchange · formal verification · LDACS

1 Introduction

Post-Quantum Cryptography The progress in quantum technology makes a potential threat of large scale universal quantum computers to cryptography undeniable. To enable secure communication in the future, quantum secure alternatives to Diffie-Hellman-like key exchanges, that do not rely on the hardness of the discrete logarithm problem, have been proposed. The most prominent of such alternatives are the post-quantum schemes that are currently undergoing standardization as part of the National Institute for Standards and Technology (NIST) post-quantum competition [fST22]. These schemes are based on key encapsulation mechanisms (KEMs), meaning that they cannot readily be combined with existing protocols that build on a Diffie-Hellman-like key exchange structure, thus invalidating security guarantees. On the other side, any protocol that achieves security based on a *generic* notion for key encapsulation mechanisms could be instantiated with any of the NIST post-quantum schemes, providing a quantum-secure protocol.

This work was supported by funding from the topic Methods for Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs. This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record of this contribution is published in Communications in Cryptology, Volume 1, Issue 1, available online at: https://cic.iacr.org/p/1/1/17.

E-mail: marcel.tiepelt@kit.edu (Marcel Tiepelt), christian.martin@kit.edu (Christian Martin), nils.maeurer@airbus.com (Nils Maeurer)

Date of this document: 2024-07-04.

Authenticated Key Exchange Authenticated key exchange enables two or more parties to secretly agree on a key, additionally assuring the parties that they are interacting with their intended peer. Common frameworks to prove the security of key agreement protocols are BR- [BFWW11] or SK-security [CK02], with adversaries that have the capabilities to send or modify any messages during protocol execution. Additionally, adversaries may either learn the initial state of a party (weak corruption), or even learn intermediate states during execution of the protocol (strong corruption). We review common protocols and their security that relate to this manuscript. A broader overview can be found in [BMS20].

The Station to Station (STS) protocol [WVOW92] combines a Diffie-Hellman (DH) key exchange with a public-key infrastructure signature scheme. At first, the initiator sends a public DH value to a responder, who generates their own DH values, computes the secret key, and responds with their own DH value and a signature on both values. The responder encrypts their return message with the shared key. In the last pass, the initiator sends their own signature on the DH public values encrypted under the shared key. A variant of the STS protocol removes the encryption of the signatures and either adds a message authentication code (MAC) on the public DH values, or includes the identity of the intended partner into the signature. This construction allows to achieve security against weak corruption.

The ISO/ IEC 11770-3 protocols, specifically ISO KAM-7 [co21, §11.7] pick up at this latter construction, but add an identifier for the intended partner to the signature and an additional message authentication code to the message flow, lifting the protocol to provide security against strong corruption.

Finally, the Internet Key Exchange (IKEv2) [KHN+14a], which is implemented in IPsec and virtually all IP-based secure communications (e.g. VPNs), follows the SIGn-and-MAc (SIGMA) approach [Kra03]. The SIGMA approach combines the public-key-infrastructure-based signature, the message authentication code and the encryption of the two using the shared key, and was proven secure against strong corruption.

The above protocols have been, originally, proven secure under the Decisional Diffie-Hellman assumption [CK02, Cre11], thus are based on only classically secure components. A first formal analysis of IKEv2 assuming a post-quantum secure variant of DH was conducted by [GGCG+21], in which case the original proof would carry over, except under the post-quantum DH assumption. An alternative approach was taken by [FKMS20], who show that IKEv2 is "post-quantum secure" if pre-shared keys are used to negotiate new key material. Further, [BBF+19] provide a detailed analysis of hybrid combiners using DH values and KEMs, along with a fine-grained analysis of adversarial power. They provide a multi-stage BR security proof for an independent authenticated key exchange based on key encapsulation mechanism from a SIGMA approach. Further, [Pei14] sketched the proof of a variant of SIGMA, which exchanged the Diffie-Hellman key exchange for a key encapsulation mechanism.

Aviation Protocols Current Air Traffic Management suffers from a lack of digitalization, for example analogue voice communications, and insufficient cybersecurity measures in aeronautical datalinks and applications. In Europe, these challenges are investigated within the Single European Sky Air traffic management Research joint undertaking [JU)23b]. As a result, a new long-range terrestrial Air-Ground communication protocol was proposed, which must include measures to "[Provide] a secure channel [...] to ensure authentication and integrity of [...] message exchange" [Aer21, Int21]. Currently, the L-band Digital Aeronautical Communications System (LDACS) [JU23a] is under standardization by the International Civil Aviation Organization [Int21], which provides a framework for air-ground communication between civilian aircraft and ground stations. A placement of LDACS in such communication networks is depicted in Figure 1.

At the heart of the LDACS security architecture lies a mutually authenticated key

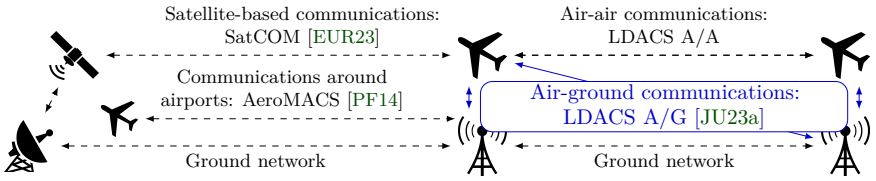

Figure 1: Placement of LDACS (solid, blue lines) in commercial aviation communication networks. AeroMACS is a short-range system to operate around airports. SatCOM allows aircraft to receive navigational support data.

exchange, instantiating either an ISO key agreement protocol [co21] with DH, or an similar protocol where the DH component is exchanged for a key encapsulation mechanism. This is motivated by the understanding, that the public key exchange schemes undergoing standardization as part of the NIST post-quantum competition [fST22] are all KEMs.

After successful key agreement, the resulting key material will be used to secure LDACS control-plane and user-plane data. Examples of user-plane data include applications such as the Controller-Pilot Data Link Communications, Automatic Dependent Surveillance-Contract or Ground-Based Augmentation System. The first is used for communications of clearances, route-changes, speed or radio frequency assignments and other mission critical data. The second application allows the automatic surveillance of the airspace and informs air traffic controllers about the position, levels and headings of aircraft. The third enables fully automatic landings, without the need of a human intervention [JU23a]. Since the landing is naturally a dangerous phase of any flight the lack of a human in the loop as a control instance requires a substantial amount of trust in these data. To the best of our knowledge previous *investigations* of the LDACS MAKE protocol (for example [MGG+21, MGS21, MGC22]) rely on heuristic arguments and do not provide security proofs or a formal analysis of the security guarantees in the computational and symbolic model.

1.1 Contribution

In this work we bridge the gap between classically and quantum secure authenticated key agreement by when using any (post-quantum) IND-CPA secure key encapsulation mechanism. Our main contribution is the analysis of the LDACS key agreement protocol in Edition 01.01.00 with Template Edition 02.00.05 from 25.04.2023 [JU23a, §C.8.2], which supports to deploys a key encapsulation mechanism instead of a Diffie-Hellman key exchange. We additionally report on the security of an alternative version of the LDACS protocol, which is detailed in Appendix A2, and which was not part of the published version [TMM24]. We provide a simplified and idealized variant of the real-world protocol in Section 3.1, which we then prove to be secure in a computational, game-based model [BFWW11, dSGFW19], by proving Theorem 1 in Section 4. Additionally, we provide an automated Tamarin proof in the symbolic model, the results of which are outlined in Section 5 assuming a Dolev-Yao attacker.

Theorem 1. *Let kem be an IND-CPA secure key encapsulation mechanism except with advantage ε_{kem} and correctness $1-\lambda_{kem}$, let sig be an EUF-CMA secure signature scheme except with advantage ε_{sig}, and prf be an ϵ_{prf} secure pseudo-random function. Let n be a bound on the number of parties and l a bound on the total number of sessions and κ a security parameter. Then the LDACS MAKE protocol (cf. Figure 3) provides (almost) full explicit entity authentication and (almost) full explicit key authentication, which implicitly includes BR-secrecy. Any quantum polynomial time bounded adversary has advantage of*

falsifying the notions of at most

$$4n\varepsilon_{sig} + \frac{4l^2}{2^{2\kappa}} + 5l\lambda_{kem} + 2l^2 \cdot \left(\frac{4(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{prf}) \right) + \frac{2}{2^{\kappa}} .$$

We provide a complete proof of Theorem 1, with modularized security notions and explicit reductions, giving the precise security loss. The protocol achieves security against polynomially bounded quantum adversaries if the respective instantiations are quantum secure, *i.e.*, if the KEM and the signatures are instantiated with post-quantum secure schemes, and the *prf* is secure with appropriate key length. Independently, we provide a model of the LDACS protocol in Tamarin, achieving security when assuming the cryptographic components to be *perfect*. The results of the latter confirm our findings for the computational proof of the LDACS protocol. We include the source code of the symbolic proof and a *Dockerfile* along with this manuscript to reproduce our results.

1.2 Technical Outline

We consider the setting of two parties each with a unique id, an initiator "ground station" *GS* and a responder "air station" *AS*, engaging in a key agreement scheme. Every party has access to a public-key infrastructure, *i.e.*, a long-term private signing key for an EUF-CMA secure signature scheme *sig* as well as a respective verification key to check signatures of an intended communication partner. Further, the parties have agreed on a cipher suite determining an IND-CPA secure key encapsulation mechanism *kem*, an EUF-CMA secure message authentication code *mac*, and a pseudo-random function *prf*.

The party *GS* initiates the protocol by generating a *fresh* KEM key pair and sending the public key pk_{kem} to *AS*. *AS* runs an encapsulation algorithm on pk_{kem} resulting in a key k and a ciphertext. A MAC key and a session key are derived from k using the pseudo-random function *prf*. Then the *AS* generates a MAC tag and a signature of the ciphertext, the unique id of *GS* and pk_{kem} using the MAC key and their long-term signature key. Finally, the *AS* sends the ciphertext, the signature and the tag to *GS*. *GS* verifies the signature, decapsulates the ciphertext, derives the MAC key and a session key and verifies the MAC tag. If the MAC and signature verifications succeed, *GS* constructs their own signature and MAC tag of the ciphertext, the id of *AS* and the KEM public key, send both to *AS* and accepts the session key. *AS* accepts their session key if the tag and the signature verify correctly. The above protocol resembles a simplification of the LDACS MAKE protocol as detailed in Section 3.1.

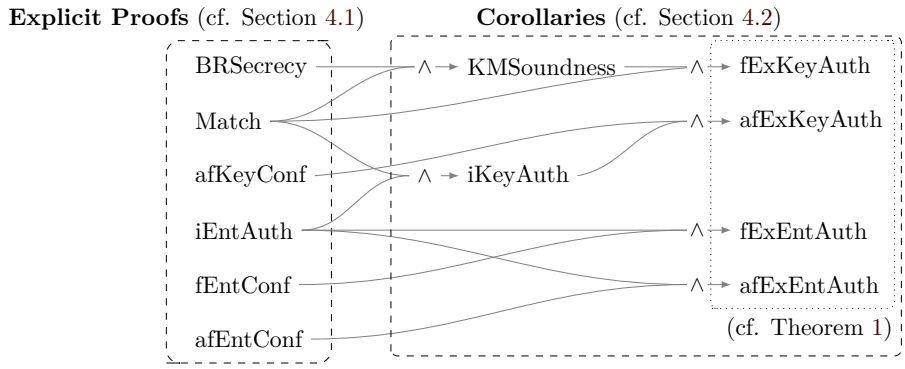

Figure 2: Overview of computational proof structure, showing how the explicit proofs of the security notions (as defined in Section 2.2) and the consequential security notions are related. The latter notions are implied by the logical conjunction of corresponding preceding notions.

The security of the simplified key agreement protocol is captured in the Match-security model [BPR00]. We adopt the predicate-based framework of [dSGFW19] to model security goals and notions. Particularly, we prove all individual notions required to assemble Theorem 1 in Section 4, achieving the following properties:

Entity Authentication. The notion of (Almost) Full Explicit Entity Authentication captures the promise that each party is assured that they interact with their intended peer, and that their peer is aware of the identity of the party. Entity authentication holds due to the EUF-CMA security of the signature scheme *sig*.

Key Authentication. (Almost) Full Explicit Key Authentication gives the parties assurance that their intended peer and only their intended peer knows the secret key, which holds due to the EUF-CMA security of *sig*, the IND-CPA security of *kem* and the pseudo-randomness of *prf*. This notion implies BR-secrecy with forward secrecy against weak corruption.

We provide a complete set of explicit proofs to various security notions of [dSGFW19] as outlined in Figure 2. The proof of BR-secrecy, which is implicitly used to assemble Theorem 1, partially follows the SK-security proof of the SIGMA protocol [Kra03, CK02], with the suggestions of [Pei14] for exchanging the DH values for a key encapsulation mechanism in the SIGMA protocol. The adaptions are marked accordingly.

Further, we provide an implementation of mutual authentication, key confirmation and secrecy corresponding to the protocol, which are proven symbolic model, the implementation of which is outlined in Section 2.3, and the results of which are detailed in Section 5. The automated proof supports the findings of the computational proof. We have made our Tamarin source code for the symbolic proof available at `https://github.com/mtiepelt/ldacs-make-symbolic-tamarin`.

Remark 1. The message authentication code used in the LDACS protocol has no impact on the security loss in Theorem 1. This is not surprising, since the KEM-based LDACS protocol features a signature similar to ISO-type protocols, which are known [Kra03, §4] to achieve security with DH-based instantiations by including the identities into the signatures. This is also in line with the main goal of LDACS to provide security against attackers that only corrupt the long-term keys.

2 Preliminaries

Let κ denote the security parameter and $x \xleftarrow{\$} X$ uniformly random sampling x from distribution X. We denote $x.y$ the (individual) y associated with x, and $\mathbb{P}[z]$ as the probability that event z occurs. Whenever we talk about an adversary $\mathcal{A}$, we assume that they are polynomially bounded.

2.1 Cryptographic Constructions

This section formally defines the various cryptographic components and their associated security notions used within the paper.

Definition 1 (Key Encapsulation Mechanism (KEM))**.** A key encapsulation mechanism *kem* is a tuple (Gen, Encaps, Decaps) of probabilistic polynomial-time (PPT) algorithms: Gen generates a secret key $\mathrm{sk} \in \mathcal{SK}$ and a public key $\mathrm{pk} \in \mathcal{PK}$, $(\mathrm{pk}, \mathrm{sk}) \leftarrow \textsc{Gen}(1^\kappa)$, Encaps generates a key $k \in \mathcal{K}$ and a ciphertext $c \in \mathcal{C}$ as $(k, c) \leftarrow \textsc{Encaps}(\mathrm{pk})$, and Decaps decapsulates the ciphertext to a shared secret, $\{k', \perp\} \leftarrow \textsc{Decaps}(\mathrm{sk}, c)$, returning either a key or an error symbol $\perp$. The KEM is λ_{kem} correct, if k' is equal to k with a probability of at least $1 - \lambda_{kem}$.

A common security notion for key encapsulation mechanism is indistinguishability (of the key) under a chosen plaintext attack (IND-CPA). In the game-based security a challenger samples a bit b^* and a key pair $(\mathrm{pk}^*, \mathrm{sk}^*)$, creates an encapsulation ciphertext c^*, and passes to the adversary pk^*, c^* and k^*, which is either the key from the encapsulation, or a random key, depending on b^*. The KEM is then ε_{kem} secure, if the advantage of distinguishing the two cases for any polynomially bounded adversary is bounded by $\varepsilon_{kem} \leq \mathit{negl}(\kappa)$, *i.e.*,

$$\mathrm{Adv}_{kem}^{\text{IND-CPA}}(\mathcal{A}, \kappa) := \mathbb{P}\left[b^* = b' | b' \leftarrow \mathcal{A}(1^\kappa, \mathrm{pk}^*, c^*, k^*)\right] \leq \varepsilon_{kem}\,.$$

Definition 2 (Signature Scheme (*sig*))**.** A signature scheme *sig* is a tuple $(\textsc{Gen}, \textsc{Sign}, \textsc{Vfy})$ of PPT algorithms: $(\mathrm{vk}, \mathrm{sk}) \leftarrow \textsc{Gen}(1^\kappa)$ samples a verification key $\mathrm{vk} \in \mathcal{VK}$ and a signature key $\mathrm{sk} \in \mathcal{SK}$. $\sigma \leftarrow \textsc{Sign}(\mathrm{sk}, m)$ uses sk to generate a signature $\sigma \in \mathcal{S}$ for some message $m \in \mathcal{M}$. $b \leftarrow \textsc{Vfy}(\mathrm{vk}, m, \sigma)$ uses vk to check whether σ is a valid signature for the message m and outputs the result as a bit $b \in \{0, 1\}$.

Similarly, in an existential unforgeability under chosen message attack (EUF-CMA) game, a challenger generates a signing key sk^* and verification key vk^* and provides the adversary with the verification key and a signing oracle. The signature scheme is then $\varepsilon_{sig} \leq \mathit{negl}(\kappa)$ EUF-CMA secure, if no polynomially bounded adversary can generate a signature, that was not previously queried to the $\textsc{Sign}$ oracle, such that the verification algorithm outputs 1, *i.e.*,

$$\mathrm{Adv}_{sig}^{\text{EUF-CMA}}(\mathcal{A}, \kappa) := \mathbb{P}\big[1 \leftarrow \textsc{Vfy}(\mathrm{vk}^*, m', \sigma') | (m', \sigma') \leftarrow \mathcal{A}^{\textsc{Sign}(\mathrm{sk}^*, \cdot)}(1^\kappa, \mathrm{vk}^*) \ \big] \leq \varepsilon_{sig}\,.$$
$$m' \textit{ not queried to } \textsc{Sign}(\mathrm{sk}^*, \cdot)$$

Pseudo-Random Function (*prf*) Pseudo-random functions are defined over a security game, where the adversary can query an oracle with an input $x \in \mathcal{X}$. The oracle returns either $\mathit{prf}(k^*, x) \in \mathcal{Y}$ for a uniformly random but fixed string k^*, or they return $\mathit{rf}(x)$ for a uniformly random but fixed function $\mathit{rf} \in \mathcal{Y}^\mathcal{X}$. The choice of output is depending on a uniformly random but fixed challenge bit. A function *prf* is then pseudo-random up to $\varepsilon_{prf} \leq \mathit{negl}(\kappa)$, if no polynomially bounded adversary can efficiently distinguish the two cases, *i.e.*,

$$\mathrm{Adv}_{prf}^{\text{PRF}}(\mathcal{A}, \kappa) := \left|\mathbb{P}\left[1 \leftarrow \mathcal{A}^{\mathit{prf}(k^*, \cdot)}(1^\kappa) \middle| k^* \xleftarrow{\$} \mathcal{K}\right] - \mathbb{P}\left[1 \leftarrow \mathcal{A}^{\mathit{rf}(\cdot)}(1^\kappa) \middle| \mathit{rf} \xleftarrow{\$} \mathcal{Y}^\mathcal{X}\right]\right| \leq \varepsilon_{prf}\,.$$

One-Way Functions (owf) An efficiently computable function $\mathit{owf}\colon \mathcal{X} \to \mathcal{Y}$ is called one-way, if for every polynomially bounded adversary and for every $\kappa \in \mathbb{N}$, the adversary cannot find a pre-image to a given, random image except with advantage $\varepsilon_{owf} \leq \mathit{negl}(\kappa)$:

$$\mathrm{Adv}_{owf}^{\text{OWF}}(\mathcal{A}, \kappa) := \mathbb{P}\left[\mathit{owf}(x^*) = \mathit{owf}(x') \mid x' \leftarrow \mathcal{A}(1^\kappa, \mathit{owf}(x^*)), x^* \leftarrow \mathcal{X}\right] \leq \varepsilon_{owf}$$

2.2 Computational Security Model

We consider game-based security for authenticated key exchange [BFWW11] as formalized by [dSGFW19], which features precise notation using predicates with explicit security properties. A short version of [dSGFW19] was published as [DFW20]. However, we refer to the full version as we believe this to be more accessible. We closely follow [dSGFW19, Sec. 2] to describe the model and its parameters.

Setup: Identities and Protocol Executions The security model is defined over a set $\mathcal{I} \subset \mathbb{N}$ of parties. Each party has a unique party identifier $i, j \in \mathcal{I}$. The total number of parties, $n = |\mathcal{I}|$, is fixed a priori. Later, we will use either GS as an identifier to indicate that a party is a ground station or AS indicating an air station, depending on the corresponding security notion achieved for a specific entity. The set of authenticating parties is $\mathcal{S}$. Since we only consider the case of mutual authentication for all parties, we have $\mathcal{S} = \mathcal{I}$. We work in the pre-specified peer model, which means, that each party is aware of their own identifier, and also knows their *intended* partner's unique identifier, which is also called the pid.

These parties engage in a protocol $\Pi = (KG, \Pi_C)$, defined by an initial key generation KG as well an algorithm Π_C corresponding to the locally executed procedure of each party. Each individual execution of a protocol by an entity $i \in \mathcal{I}$ is called a local session, associated with a local session identifier $\ell = (i, j, k)$ where k denotes that this is the k-th session between party i and party j. The total number l of local sessions is bound a priori, such that $k \leq l$.

Game States and Bookkeeping The security of a protocol Π is defined via an experiment called a *game*, where the adversary engages in protocol executions and adaptively interacts with or corrupts parties. [dSGFW19] introduce five lists to track the individual states of protocol executions and the game.

1. The list SST of protocol-related session states stores for each local session $\ell = (i, j, k)$ a signature key pair $(\mathrm{vk}_i^{sig}, \mathrm{sk}_i^{sig})$ of party i as well as their intended partner's verification key vk_j^{sig}. Further, it stores the session identifier, key and two variables, $\mathrm{sid}, \mathrm{K}, \mathrm{kcid}, \mathrm{ecid} \in \{0,1\}^* \cup \perp$, all of which are initialized to $\perp$, and are set as soon as the local session accepts. The latter two are set during a session and then remain invariant for the remainder of the session. The kcid's purpose is to unambiguously determine the value of the session key before the session accepts while the ecid determines the value of the session identifier. The model features a key confirmation flag $kconf \in \{\mathrm{none}, \mathrm{almost}, \mathrm{full}\}$, denoting which form of confirmation should be achieved. For GS the flag is set to "almost", and for the AS to "full".

2. The list LST of game-related local session states corresponds to how the adversary interacts with the sessions throughout the game. For each session, it stores whether the owner and peer of a session are honest or corrupted, $\delta_{\mathrm{ownr}}, \delta_{\mathrm{peer}} \in \{\mathrm{honest}, \mathrm{corrupt}\}$, and similarly the state of a session $\delta_{\mathrm{sess}} \in \{\mathrm{fresh}, \mathrm{revealed}\}$. Parties and sessions are honest and fresh by default, and may change their state depending on the adversary's interaction.

3. The list corresponding to the game execution state EST contains information, $\{(i, \mathrm{vk}_i^{sig}, \mathrm{sk}_i^{sig}, \delta_i)\}_i$, denoting which parties have been corrupted (independently of sessions).

4. LSID is the list of valid local sessions identifiers.

5. The list MST holds information related to specific security notions.

Experiment and Adversarial Interaction Security is defined via an experiment where an adversary interacts with a game. In the setting of quantum-secure communication the adversary $\mathcal{A}$ is a quantum polynomial-time algorithm which interacts through classical queries with the game. The set of all queries is $Q = \{\textsc{Send}, \textsc{Reveal}, \textsc{Corrupt}\}$. On input $\textsc{Send}(\ell, m)$, the game processes Π_C on m on behalf of local session ℓ, and returns the result to the adversary. On input $\textsc{Reveal}(\ell)$, the game sets $\ell.\delta_{\mathrm{sess}}$ to revealed, and returns $\ell.\mathrm{K}$ to the adversary. On input $\textsc{Corrupt}(i)$, the entity i is marked as corrupt in

variable δ_i. Then, for any session owned by that entity $(i, \cdot, \cdot)$, and any session where it is listed as a peer $(\cdot, i, \cdot)$, the entity is marked as corrupt, *i.e.*, the game sets $\delta_{\text{ownr}} = \text{corrupt}$ (respectively for δ_{peer}). Finally, the secret signing key sk_i^{sig} is returned to the adversary.

The game returns responses to the adversary according to an algorithm χ, which evaluates a query $q \in Q$ and the game state. [dSGFW19] further allow the adversary to submit invalid queries, determined by a predicate *Valid*, for which χ is not executed. We note that such queries are not relevant in our setting. The state of each game G comprises the five lists from the previous paragraph.

The experiment then consists of three phases: In a first phase of the experiment, key pairs for all entities are generated and variables initialized as previously defined. In the second phase, the adversary may interact with the sessions and parties by submitting queries q to the game, which are processed by χ. In the last phase, after the adversary send all their queries and terminated, the game evaluates a predicate on the state $b \leftarrow P(\text{SST}, \text{LST}, \text{EST}, \text{LSID}, \text{MST})$ and outputs the bit $b \in \{0, 1\}$. The adversary wins the game, if $b = 0$.

With this setup, a security notion is fully defined by a predicate P. A protocol achieves a security property if for all polynomial-time adversaries $\mathcal{A}$ the predicate P evaluates to 1 except with negligible probability, *i.e.*,

$$\text{Adv}^G_{\mathcal{A},\Pi,\mathcal{I},\mathcal{S}} = \mathbb{P}\left[\text{Exp}^G_{\mathcal{A},\Pi,\mathcal{I},\mathcal{S}}(1^\kappa) = 0\right] \leq \mathit{negl}(\kappa)\,, \tag{1}$$

where $\text{Exp}^G_{\mathcal{A},\Pi,\mathcal{I},\mathcal{S}}$ corresponds to running the game G with protocol Π, parties $\mathcal{I}$ and authenticating parties $\mathcal{S}$ against the adversary $\mathcal{A}$. We note that for the remaining paper we only write "adversary" instead of "polynomial-time adversary" for simplification, and security holds against quantum polynomial-time adversaries if the respective components are post-quantum secure.

2.2.1 Predicates

We recall the definitions of the various security properties and the predicates introduced by [dSGFW19] with the simplifications relevant to this paper. Note that all predicates are quantified over the list of valid local session identifiers, *i.e.*, $\forall \ell \in \text{LSID}$. The predicates SameKey, SameKCID, SameECID and Partnered are defined to enable the readability of the subsequent predicates:

Definition 3 (SameKey)**.** *cf.* [dSGFW19, Def 2.3]. $\text{SameKey}(\ell, \ell') \Leftrightarrow \ell.\text{K} = \ell'.\text{K} \neq \perp$ for distinct sessions ℓ, ℓ'. Remark: The definitions for SameKCID and SameECID are analogous.

Definition 4 (Partnered)**.** *cf.* [dSGFW19, Def 2.1]. $\text{Partnered}(\ell, \ell') \Leftrightarrow \ell \neq \ell'$ and $\ell.\text{sid} = \ell'.\text{sid} \neq \perp$.

The Match property promises, intuitively, that two parties engaging with the same session identifier also derive the same key K and kcid, and that the latter guarantees the computation of identical keys. Additionally, the property promises that each local session is partnered with at most one other local session.

Definition 5 (Match predicate)**.** *cf.* [dSGFW19, Def 2.3].

$$\begin{aligned}\text{Match} \Leftrightarrow \forall \ell, \ell', \ell'' : &(\text{Partnered}(\ell, \ell') \wedge \ell.\text{K} \neq \perp \neq \ell'.\text{K}) \Longrightarrow \text{SameKey}(\ell, \ell') \\ &\wedge (\text{Partnered}(\ell, \ell') \wedge\ \ell.\text{kcid} \neq \perp \neq \ell'.\text{kcid}) \Longrightarrow \text{SameKCID}(\ell, \ell') \\ &\wedge (\text{Partnered}(\ell, \ell') \wedge\ \text{Partnered}(\ell, \ell'')) \Longrightarrow \ell' = \ell'' \\ &\wedge (\text{SameKCID}(\ell, \ell') \wedge \ell.\text{K} \neq \perp \neq \ell'.\text{K}) \Longrightarrow \text{SameKey}(\ell, \ell')\end{aligned}$$

Entity and Key Authentication Implicit entity authentication holds, if any party is guaranteed that for each of its sessions, only its intended partner has a matching sid.

Definition 6 (Implicit Entity Authentication)**.** *cf.* [dSGFW19, Def 3.1].

$$\textsc{iEntAuth} \Leftrightarrow \forall \ell \text{ that accept} : (\forall \ell' : \textsc{Partnered}(\ell, \ell') : \ell'.\text{id} = \ell.\text{pid})$$

Entity confirmation improves this by providing assurance that there exists another session with the same sid ("full"), or the same ecid ("almost full"), where for the latter matching ecids eventually result in the same sid if the respective session terminates successfully.

Definition 7 (Full Entity Confirmation)**.** cf. [dSGFW19, §9.1]

$$\textsc{fEntConf} \Leftrightarrow \forall \ell \text{ that accept} \wedge \ell.\delta_{\text{peer}} = \text{honest} : \exists \ell' : \textsc{Partnered}(\ell, \ell')$$

Definition 8 (Almost-Full Entity Confirmation)**.** *as suggested in* [dSGFW19, §9.1].

$$\textsc{afEntConf} \Leftrightarrow \forall\ \ell \text{ that accept} \wedge \ell.\delta_{\text{peer}} = \text{honest} : \exists \ell' : (\textsc{SameECID}(\ell, \ell') \wedge (\ell'.\text{sid} \neq \perp \implies \textsc{Partnered}(\ell, \ell')))$$

"Almost" full key confirmation promises, that parties engaging with an honest peer have assurance, that their peer derives the same key. We note that "full" key confirmation is implied by another, stronger predicates later on, and thus not explicitly defined.

Definition 9 (Almost-Full Key Confirmation)**.** cf. [dSGFW19, Def 3.4]

$$\textsc{afKeyConf} \Leftrightarrow \forall \ell \text{ that accept} \wedge \ell.\delta_{\text{peer}} = \text{honest} : \exists \ell' : \textsc{SameKCID}(\ell, \ell') \wedge ((\ell'.\text{K} \neq \perp) \Rightarrow \textsc{SameKey}(\ell, \ell'))$$

The following four primary security notions of [dSGFW19] are the main terms in our Theorem 1. (Almost) Full explicit entity authentication promises that for honest, authenticating parties only the intended partner has a matching sid and there exists at least one such session with matching sid (respectively ecid for "almost").

Definition 10 (Full Explicit Entity Authentication)**.** *cf.* [dSGFW19, §9.1]

$$\textsc{fexEntAuth} \Leftrightarrow \forall \ell \text{ that accept} : (\forall \ell' : \textsc{Partnered}(\ell, \ell') : \ell'.\text{id} = \ell.\text{pid}) \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' : \textsc{Partnered}(\ell, \ell'))$$

Definition 11 (Almost-Full Explicit Entity Authentication)**.** *cf.* [dSGFW19, §9.1]

$$\textsc{afexEntAuth} \Leftrightarrow \forall \ell \text{ that accept} : (\forall \ell' : \textsc{Partnered}(\ell, \ell') : \ell'.\text{id} = \ell.\text{pid}) \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' : \textsc{SameECID}(\ell, \ell') \wedge (\ell'.\text{sid} \neq \perp \implies \textsc{Partnered}(\ell, \ell')))$$

Similarly, (almost) full explicit key authentication promises that only the intended partner has sessions with matching key and that at least one such session with matching key exists (respectively kcid).

Definition 12 (Full Explicit Key Authentication)**.** cf. [dSGFW19, Def 3.5]

$$\textsc{fexKeyAuth} \Leftrightarrow \forall \ell \text{ that accept} : (\forall \ell' : \textsc{SameKey}(\ell, \ell') : \ell'.\text{id} = \ell.\text{pid}) \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' \text{ such that } \textsc{SameKey}(\ell, \ell'))$$

Definition 13 (Almost-Full Explicit Key Authentication)**.** cf. [dSGFW19, Def 3.6]

$$\textsc{afexKeyAuth} \Leftrightarrow \forall \ell \text{ that accept} : (\forall \ell' : \textsc{SameKey}(\ell, \ell') : \ell'.\text{id} = \ell.\text{pid}) \wedge (\ell.\delta_{\text{peer}} = \text{honest} \implies \exists \ell' : \textsc{SameKCID}(\ell, \ell') \wedge (\ell'.\text{K} \neq \perp \implies \textsc{SameKey}(\ell, \ell')))$$

Secrecy To define BRSec, [dSGFW19, §5] provide a modified game G_{BRSec}, where the adversary's goal is to distinguish a real and a random session key. The modified game state includes EST, SST, LST. The state MST includes two additional bits and a challenge session. The bit b_{test} holds whether the adversary is provided with a real key from a session, or whether they are provided with a random key. The bit b_{guess} will hold the adversary's guess. The challenge session ℓ_{test} will correspond to the adversary's choice of which session to test. To set these states, the game provides two additional queries to the adversary. On input $\textsc{Test}(\ell)$, the game sets the test session $\ell_{\text{test}} \leftarrow \ell$ and returns $\text{K} = \ell.\text{K}$ if $b_{\text{test}} = 1$, or a random key K from the key space if $b_{\text{test}} = 0$. On input $\textsc{Guess}(b)$, the bit $b_{\text{guess}} \leftarrow b$ is updated. Additionally, the game restricts the adversary to submit only a single Test query. The the BRSec predicate evaluates to the bit b_{guess} as in Definition 14:

Definition 14 (BR-Secrecy)**.** [dSGFW19, Def 5.1, 5.2] The BRSec predicate is defined as follows:

$$\begin{aligned}
\textbf{if} \quad & \text{MST}.\ell_{\text{test}} \neq \perp \wedge \ell.\delta_{\text{ownr}} = \ell.\delta_{\text{peer}} = \text{honest} \wedge \ell.\delta_{\text{sess}} = \text{fresh} \\
& \wedge \forall \ell' \in \text{LST} : \textsc{Partnered}(\ell, \ell') \Rightarrow \ell'.\delta_{\text{sess}} = \text{fresh} \\
\textbf{then} \quad & \text{BRSec} \leftarrow \text{MST}.b_{\text{guess}} \\
\textbf{else} \quad & \text{BRSec} \leftarrow 0
\end{aligned}$$

Let $G^{b_{\text{test}}=b}_{\text{BRSec}}$ denote the BR-secrecy game during which b_{test} has the value b. Due to the distinguishing nature of the BR-secrecy game, the adversarial advantage is defined as

$$\text{Adv}^{G_{\text{BRSec}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) := \left| \mathbb{P}\left[\text{Exp}^{G^{b_{\text{test}}=0}_{\text{BRSec}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) = 1\right] - \mathbb{P}\left[\text{Exp}^{G^{b_{\text{test}}=1}_{\text{BRSec}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) = 1\right] \right| .$$

Remark 2. In [dSGFW19, Def 5.2], the BRSec predicate has value $\perp$ in some cases. Since the adversarial advantage does not distinguish whether BRSec has the value 0 or $\perp$, we omit this detail.

2.3 Symbolic Security Model

The symbolic proof model establishes the correctness and security of a cryptographic protocol without making assumptions about the computational limitations of an adversary. The adversary is modeled as a Dolev-Yao attacker [NS78, DY83], who can send, receive or alter messages in interactive protocols, and who has access to cryptographic primitives as perfect black-boxes. The messages sent over the network are inputs to these black-boxes and the adversary is restricted to use only these. A protocol is formalized using correspondence assertions of the form "If event x is executed, then event y is executed". An automated symbolic prover, such as Tamarin [MSCB13], can check whether a protocol fulfills security goals by (exhaustively) exploring the possible options for parties (called agents) assuming the various roles in the protocol.

In Tamarin [MSCB13], security is formalized over a system state as an initially empty multi-set of predicates called *facts*. *Rules* define how the state can transition to a new set, adding ore removing facts from the system state. Each rule is associated with a premise and a conclusion. A rule can only be applied, if all facts of the premise are present in the system state. If the rule is applied, then the system state is updated according to the facts in the conclusion. Additionally, rules may features action facts, which record the application of the rule by appending all action facts to a trace.

With these rules the algorithmic behaviors of a protocol can be modeled by defining rules mimicking the interaction of the agents. Similarly, adversarial powers can be modeled via action facts, for example, a rule may feature an action fact representing that a long-term secret is available to the adversary and that the corresponding party is corrupted.

Security properties on the other hand are modeled via traces of action facts. These are denoted lemmas. This means, a property is modeled by a giving a formula which is evaluated on the traces. Then a security property can be modeled either with an existential ("exists") or an universal ("all") quantification, indicating that the formula has to evaluate to *True* either for any one trace, or for all traces. Usually, security properties are required to hold for all traces.

Protocol and Adversarial Rules The Tamarin manual [Tea23] already provides built-in code for signing and hashing, Celi et al. [CHSW22a, §3.2.1] provide a Tamarin instantiation of a KEM, which satisfies Definition 1. The protocol described in Section 2.3 is modeled via rules and facts, allowing each party corresponding to a unique identifier to register a long-term key, and then to engage with any other party in a protocol session.

To model the adversary additional rules are added that allow to reveal the long-term key, a KEM key or a session key: The rule "Reveal_ltk" has as premises a long-term key associated with an agent X, an action fact that marks the agent as corrupted via the fact "CorruptedLtk(X)", and which adds the long-term key in question as a conclusion fact to the state. Analogously, "Reveal_kem" and "Leak_session" allow to reveal the secret key output by Encaps, Decaps, or the session key. As such, the adversary in the symbolic model is slightly stronger than in the computational model, as they are allowed to have the KEM key leaked (in the computational model, only long-term key via a *Corrupt* query and session keys via *Reveal* queries (or *Test* queries, in the BRSec predicate)).

Security Properties In the following we describe the lemmas associated with the security properties relevant to this work, where agent A and B correspond to the ground- and air stations:

mutual_authentication_A/B The property is modeled via Lowes [Low97] bi-directional full-agreement property as outlined in the Tamarin manual [Tea23], combined with a unique element exchanged between agent A and B. Full agreement is the combination of injective agreement, *i.e.*, stating that if agent A accepts with agent B, then they can be sure B also accepted with A, except if either of long-term keys was revealed (resulting in corruption of the agent as an action fact). The uniqueness property is modeled via the *session_ uniqueness_A/B* lemma (modeled as in [GGCG+21]), *i.e.*, the guarantee that different sessions have different keys. When both lemmas hold, mutual authentication is achieved via full-agreement. Hence, the *mutual_authentication_A/B* lemma can be regarded as a combination of Definitions 6 to 8.

secrecy Secrecy corresponds to BR-secrecy, promising that when a session key x is assigned a "secret" property at time i, then either the adversary does not know x, or x has been revealed, or a corresponding agent was marked as corrupted via an action fact. Similarly, *secrecy_pfs* guarantees, that the adversary does not know x, except if the agent was corrupted at a time j prior to i, *i.e.*, $j < i$.

key_consistency_A/B Key consistency [GGCG+21, §A.6.4] corresponds to key confirmation, such that for all sessions with agents A and B with keys keyA and keyB respectively, the following holds: When agent A accepts keyA at time i in session i_A, and agent B accepts with keyB at time j in the same session, then keyA and keyB must be same, except if one of the agents was corrupted. This corresponds to key confirmation in the computational model.

The models of session uniqueness and key confirmation are based on the implementation from Donenfeld et al. [DM17].

3 LDACS

In this section, we provide a simplified protocol resembling the LDACS MAKE. The LDACS MAKE protocol in Edition 01.01.00 with Template Edition 02.00.05 from 25.04.2023 [JU23a, §C.8.2] features two individual definitions that either use a Diffie-Hellman key exchange, or a KEM based key exchange. The first instantiates the standardized *Key Agreement Mechanism 7* (KAM-7) [co21, §11.7], and inherits all security properties. As such, we do not provide any further analysis of this instantiation. The latter substitutes the Diffie-Hellman key exchange for a construction using a key encapsulation mechanism, and thus does not instantiate ISO KAM-7, as this requires to be instantiated with a commutative function F: The session key is computed as $k = F(r_i, F(r_j, g)) = F(r_j F(r_i, g))$ [co21, §10.2], where r_i is a random value provided by one party, and r_j by the other party. The variant with the key encapsulation, however, computes the session key from $k, c \leftarrow \textsc{Encaps}(\mathrm{pk}^i_{kem}, r_j)$ and $k \leftarrow \textsc{Decaps}(c)$, and thus does strictly not instantiate ISO KAM-7. As a consequence, the KEM-based protocol does not inherit the security properties of the ISO KAM-7. An examination of the security objectives [oI23, JU23a], which are in understanding of the official security requirements for aeronautical communications set by the International Civil Aviation Organization (ICAO) [Aer21, Int21], is provided in this context. In Section 3.2 we compare our results to other authenticated key agreements.

3.1 Key Agreement Protocol

The LDACS MAKE protocol is split into two phases: In the *cell entry* stage, the two parties ground station GS and air station AS exchange miscellaneous information over an insecure channel with no attempt to achieve any security. At the end of the cell entry, both parties are aware of an intended partner associated by an identifier id_{GS} and id_{AS}, as well as a cipher suite used in the instantiation of the subsequent stage. Additionally, the AS may send a flag signaling that they do not have the GSs signature public key vk^{sig}_{GS}, which may then be sent (in plain) by GS. We note that no security must hold for this initial phase, and this is not modeled in the protocol. Instead, we assume the intended entity identifiers id_{GS}, id_{AS} and respective signature keys from the public key infrastructure to be an input to the respective party. In the second stage, the two parties engage in a mutual authenticated key agreement protocol to exchange a secret that is used to derive multiple keys. This second stage is the 3-round MAKE protocol between two parties, the KEM-variant of which is the focus our analysis.

The protocol described in Figure 3 is a simplifications of the second stage from Section 3.1: Both parties take as input a unique identifier pid for an intended partner, their own unique identifier id, the public key $\mathrm{vk}^{sig}_{\mathrm{pid}}$ of the intended partner, as well as the secret signing key $\mathrm{sk}^{sig}_{\mathrm{id}}$, the latter two of which are provided by the public key infrastructure. After completing the protocol, if the parties accept, then they have a session key K and assurance that only their intended partner has knowledge of this key. The desired security properties are discussed below, the achieved security notions along with the respective proof in Section 4. We prove the computational security of these notions in the game-based model [BFWW11, DFW20] for authenticated key exchange, and verify the security properties in the symbolic model using the Tamarin prover.

Simplifications The LDACS MAKE protocol in the specification [JU23a, §C.8.2] details the instantiation of the key derivation as well as identifier strings constructed from *constant* values that result in multiple individual session keys, which are not relevant to security. In order to de-clutter the specification and provide a clear view of the protocol in question, we simplified the specification. In this paragraph we make these simplifications explicit and provide assumptions and justifications required for security of the protocol:

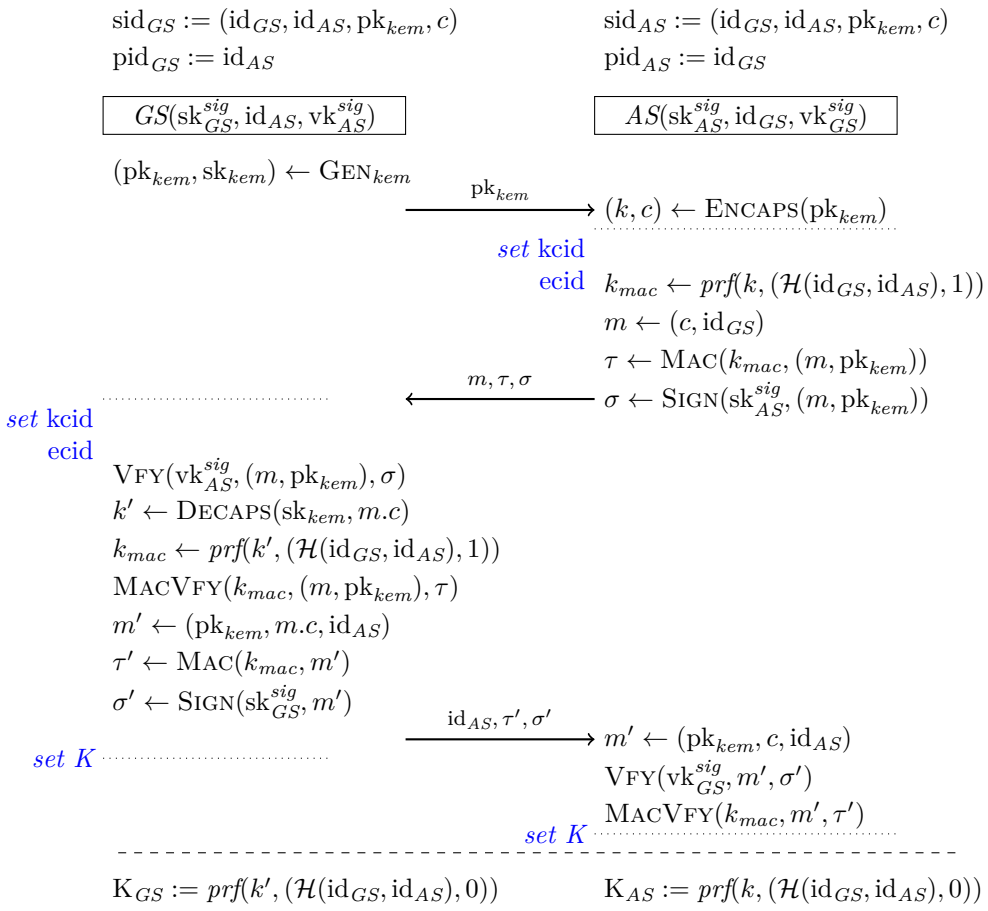

Figure 3: Simplified key agreement protocol between the ground station *GS* and the air station *AS*. We denote this protocol LDACS MAKE. If any of the verification VFY or MACVFY fail, the respective party aborts. The dotted lines with the blue identifiers are relevant for the computational proof in Section 4.

(1) The ground and air station are uniquely identified by $\mathrm{id}_{GS} = (\mathrm{UA}_{GS}, \mathrm{SAC}_{GS})$ and $\mathrm{id}_{AS} = (\mathrm{UA}_{AS}, \mathrm{SAC}_{AS})$ respectively.

(2) If $\mathrm{id}_{GS}, \mathrm{id}_{AS}$ are unique identifiers, then so is $\mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS})$.

(3) Security achieved for the session key K_X also holds for multiple keys, *i.e.*, "DCH session key", "Voice session key", "DCCH session key" and "Key encryption key".

(4) The composition of *HKDF* and $HMAC_{Hash}$ is a pseudo random function.

(5) Authentication Centers (AuC) are treated as if they were *GS*s.

Simplification (1) is purely cosmetic. Simplification (2) is justified because *SHA256*, $\mathcal{H}$, is believed to be a collision resistant hash function [fST15]. Simplification (3) is justified because all keys are derived in the same manner, by computing $prf(k, \mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS}))$ with constant strings as additional inputs. Simplification (4) is justified, because *HMAC* instantiated with *SHA256* is a pseudo-random function as shown by [Bel06]. This assumes that *HKDF* is instantiated accordingly. For Simplification (5), the specification mentions the possibility for *GS* to verify the *AS* certificates by communicating with an *Authentication*

Center (AuC) over a secure channel [JU23a, §C.8.2] (see the "dashed" box). Since the communication with the AuC is considered secure we ignore this special case, and simply assume that the ground station can verify the correctness of the AS certificate locally. To the best of our knowledge our simplifications do not invalidate any security guarantees for the protocol as in the specification, since we only merge redundant components.

Security Goals Aeronautical standards are split in two parts: First, the ICAO Standards And Recommended Practices (SARPs) define requirements of the technology; second, the specification described the technical realization of these requirements. As such, LDACS security goals are described in the SARPs [oI23], with the LDACS specification [JU23a] detailing the protocols and other security measures.

LDACS SARPs [oI23] set requirements for providing integrity protection [oI23, §13.8.2] and confidentiality [oI23, §13.8.4] of messages in transit, as well as mutual authentication [oI23, §13.8.5] between ground and air stations. These basic requirements were extended upon in the LDACS specification, *i.e.*, that the key agreement fulfills mutual authentication and key establishment [JU23a, §C.8.2], as well as key confirmation [JU23a, §C.2, table 96] without specifying details. Further, the requirements ask for a "capability to prevent the propagation of intrusions within the LDACS access networks" [oI23, §13.8.8], which may be interpreted as forward secrecy, such that leakage of long-term keys does not invalidate security properties of other, honest parties.

3.2 Comparison with other KEM-based Key Agreements

SIGMA-style Authentication The simplified protocol in Figure 3 is similar to the protocol suggested by [Pei14, §6.2], with adjustments to the messages used to generate the signature and the tag. [Pei14, Thm 6.1] states, that their protocol is secure in the CK02 model but no complete proof is provided. Indeed, our proof of BR-secrecy partially follows the sketch they outlined, and we mark this in Section 4 where appropriate.

[BBF+19] present a compiler for hybrid authenticated key exchange, combining multiple KEMs and a SIGMA-style authentication scheme [BBF+19, Fig. 13]. Particularly, their scheme exchanges a KEM public key and ciphertext, which is subsequently authenticated along with a randomly sampled string (a nonce) along with the party identifiers. The authors show that their AKE construction provides BR-Match security and BR key secrecy in the BR93 model if the KEM is IND-CPA secure and if the signature scheme and the MAC are EUF-CMA secure, thus providing security against quantum adversaries, if the underlying components are quantum secure. In contrast, the protocol used in LDACS as well as our abstraction in Figure 3 uses ISO-style authentication and no nonces.

Comparison to Key Exchange in IKEv2 The minimal key exchange component [KHN+14b, Section 1.2] of the *Internet Key Exchange Version 2* [KHN+14a] consists of an initialization step IKE_SA_INIT and an authentication step IKE_AUTH, corresponding to the first three messages exchanged between the parties. Similarly to our setting, IKEv2 builds on an existing public key infrastructure and assumes that both parties can validate respective signatures. In the first step, the two parties perform a DH key agreement along with exchanging a randomly chosen nonce for each party. Using the shared secret and the two nonces they derive several keys [KHN+14a, Section 2.14] such as a session key and an additional key used for further authentication. In the second step, IKE_AUTH, each party sends an authenticated encryption with associated data (AEAD) of its own id and of a signature. The signature contains the message the party sent in the IKE_SA_INIT exchange, the nonce of the other party and the output of a PRF with the id of the party as input [KHN+14a, Section 2.15]. The AEAD and the PRF use some of the keys derived after IKE_SA_INIT.

The main differences between the key agreement of IKEv2 and the simplified protocol Figure 3 are that IKEv2 uses DH specifically and makes use of additional nonces. Further, they use an AEAD for the IKE_AUTH messages, and the signatures contain neither the id nor the DH exponential of the respective other party. They use a PRF instead of a MAC, the PRF output is contained in the signature and the PRF does not take the id of the other party as input. We briefly review the changes in the proof required to be applicable to IKEv2 in Appendix A1.2. Finally, IKEv2 [KHN+14a, §2.12] allows the reuse of DH exponents. Since the KEM's pk and c in the LDACS protocol also function as nonces (*i.e.*, for each session they are chosen *freshly* uniformly at random and are unique except with negligible probability), it is essential that they are not reused, which is in line the LDACS specification.

4 Computational Proof

In this section we present the computational proof of security for the key agreement protocol described in Section 3. Specifically, we prove correctness of Implicit Entity Authentication (cf. Theorem 3) and BR secrecy (cf. Theorem 7) for both parties, Full entity confirmation (cf. Theorem 4) for the air station AS, and Almost Full Entity Confirmation (cf. Theorem 5) and Almost Full Key confirmation (cf. Theorem 6) for the ground station GS. From this, additional properties can be inferred, as detailed in Section 4.2.

4.1 Explicit Proofs and Reductions

We start by proving Lemma 1 which will be used throughout the subsequent proof. For all of the proof we set $\text{kcid} = \text{sid} = \text{ecid} = (\text{id}_{GS}, \text{id}_{AS}, \text{pk}_{kem}, c)$. For the remaining section we note that the key output space of the *kem* and the key input space of the *prf* are identical. The same holds for the key space of the *mac* and the co-domain of the *prf*, *i.e.*,

$$\mathcal{K}_{kem} = \mathcal{K}_{prf}$$
$$\mathcal{Y}_{prf} = \mathcal{K}_{mac}\,.$$

Lemma 1. *Let $\mathcal{A}$ be an adversary interacting with the LDACS MAKE protocol. Let sig be an ε_{sig} secure signature scheme and n the number of parties participating in the protocol. Then the probability that an adversary can forge a signature of a party P that was not corrupted (cf. Definition 21) is bounded by*

$$\mathbb{P}\left[\textsc{Sig-Forge}\right] \leq n \cdot \varepsilon_{sig}\,.$$

Proof. For an adversary forging a signature we can construct a trivial reduction that first guesses the impersonated party, and then uses the forgery to win the EUF-CMA game, thus the lemma only depends on the security of the signature scheme. A full proof is given in Appendix A1, which loosely corresponds to part of [CK02, Lemma 7]. □

Theorem 2 (Match Security, cf. Definition 5)**.** *Let kem be λ_{kem} correct and l the number of local sessions. Any adversary's advantage to falsify the MATCH predicate in LDACS MAKE is bounded by*

$$\forall \text{ adversaries } \mathcal{A} : \text{Adv}^{G_{\textsc{Match}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq \frac{l^2}{2^{2\kappa}} + l\lambda_{kem}\,.$$

Proof. Recall that the kcid and session identifier are identical, $\text{kcid} = \text{sid}$, thus if ℓ, ℓ' are partnered and the kcid is set, then they also Match. Since $\ell.\text{kcid} := (\text{id}_{GS}, \text{id}_{AS}, \text{pk}_{kem}, c)$, they have the same view of the ciphertext c and the public key pk_{kem}, and thus the keys Match, except with probability λ_{kem}, the latter of which can occur on every session in

question. Second, if the sessions ℓ', ℓ'' both share the same view of the sid with session ℓ, then $\ell' = \ell''$ except if there is a collision in both the public key and ciphertext, which happens with probability at most $1/2^{2\kappa}$ for each pair of instances. □

Theorem 3 (Implicit Entity Authentication, cf. Definition 6)**.** *Any adversary's advantage of falsifying the* iEntAuth *predicate is zero:*

$$\forall \text{ adversaries } \mathcal{A} : \mathrm{Adv}^{G_{\text{iEntAuth}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) = 0$$

Proof. This is trivially fulfilled due to Matching sid $:= (\mathrm{id}_{GS}, \mathrm{id}_{AS}, \mathrm{pk}_{kem}, c)$. □

Theorem 4 (Full Entity Confirmation, cf. Definition 7)**.** *Let sig be an ε_{sig} secure signature scheme. Any adversary's advantage of falsifying the* fEntConf *predicate for the air station AS is bounded by*

$$\forall \text{ adversaries } \mathcal{A} : \mathrm{Adv}^{G_{\text{fEntConf}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq n\varepsilon_{sig}$$

Proof. Let ℓ_{AS} be a session with peer ℓ_{GS}. Then ℓ_{AS} accepts after they received a signature $\sigma \leftarrow \textsc{Sign}(\mathrm{sk}^{sig}_{GS}, (\mathrm{id}_{AS}, c, \mathrm{pk}_{kem}))$ such that $\textsc{Vfy}(\mathrm{vk}^{sig}_{GS}, (\mathrm{id}_{AS}, c, \mathrm{pk}_{kem}), \sigma)$ evaluates to 1. For an honest peer $\ell_{AS}.\mathrm{pid} = \ell_{GS}.\mathrm{id}$ it holds that $\ell_{GS}.\mathrm{sid} := (\ell_{GS}.\mathrm{id}, \ell_{AS}.\mathrm{id}, \mathrm{pk}_{kem}, c)$, which is the same view of sid that ℓ_{AS} has, and thus they are partnered. For the successful verification we can construct a reduction to the EUF-CMA security of the signature analog to that of Lemma 1, such that the advantage is bounded by $n\varepsilon_{sig}$.

Remark: This does not hold for GS, since GS accepts after sending the last message, and AS may not have received the message yet, thus not set their sid *yet.* □

Theorem 5 (Almost-Full Entity Confirmation, cf. Definition 8)**.** *Let sig be an ε_{sig} secure signature scheme. The* afEntConf *predicate is fulfilled for the ground station GS if sig is a secure signature scheme:*

$$\forall \text{ adversaries } \mathcal{A} : \mathrm{Adv}^{G_{\text{afEntConf}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq n\varepsilon_{sig}$$

Proof. The proof is the same as for Theorem 4 with the argument over the ecid instead of the sid. A *GS* accepts only, if they receive a signature $\sigma \leftarrow \textsc{Sign}(\mathrm{sk}^{sig}_{AS}, (c, GS.\mathrm{sid}, \mathrm{pk}^{kem}_{GS}))$ such that $\textsc{Vfy}(\mathrm{pk}^{sig}_{AS}, \sigma)$ evaluates to 1. This means, that if the signature was generated by an honest *AS*, then $AS.\mathrm{ecid} := (AS.\mathrm{id}, GS'.\mathrm{id}, c, \mathrm{pk}^{kem,AS})$, which is the same as that of the *GS*. Since the ecid=sid, if the $AS.\mathrm{sid}$ is ever $\neq \perp$, then they have the same sid and are thus partnered. If the signature was not generated by an honest *AS*, we can construct a reduction to the EUF-CMA security of the signature analog to that of Lemma 1, such that the advantage is bounded by $\mathbb{P}[\textsc{Sig-Forge}] \leq n \cdot \varepsilon_{sig}$. □

Theorem 6 (Almost-Full Key Confirmation, cf. Definition 9)**.** *Let sig be an ε_{sig} secure signature scheme, and kem λ_{kem} correct. The* afKeyConf *predicate is fulfilled for the ground station GS, if sig is a secure signature scheme:*

$$\forall \text{ adversaries } \mathcal{A} : \mathrm{Adv}^{G_{\text{afKeyConf}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq n\varepsilon_{sig} + l\lambda_{kem}$$

Proof. The proof is the same as for Theorem 4, but with kcid instead of sid. Additionally, afKeyConf requires ℓ, ℓ' to have the same key, which is the case if the kcids Match, except if the *kem* fails to decapsulate correctly. □

Theorem 7 (BR-Secrecy, cf. Definition 14)**.** *The* BRSec *predicate is fulfilled for both parties, if kem, prf and sig are secure, i.e., $\varepsilon_{kem}, \varepsilon_{prf}, \varepsilon_{sig} \leq negl(\kappa)$:*

$$\forall \text{ adversaries } \mathcal{A} : \mathrm{Adv}^{G_{\text{BRSec}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq \frac{4(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{prf})$$

Proof. The proof of BR-secrecy for the LDACS MAKE protocol shows that the probability of an adversary to distinguish the experiment G_{BRSEC} outputting a real or random key is negligible. In this section we sketch the proof, a full proof can be found in Appendix A1.1.

We follow the proof of SK-security for the basic SIGMA protocol [CK02], which defines five hybrids via five variants of a simulator $\hat{S}$. Each of the $\hat{S}$ variants guesses the test session and its partner in order to modify the generation of the session and mac keys for these two local sessions. If $\hat{S}$ guesses incorrectly or the adversary manipulates the communication between the two local sessions, then the simulator aborts. The first of the hybrids ("REAL") corresponds to the G_{BRSEC} game with real TEST output, unless the associated simulator $\hat{S}_{\text{REAL}}$ aborts. In the second, denoted "RPRF", the shared secret is exchanged for randomness which is used to compute the session and mac key. In the third game, "ALLR", both the session and the mac key are exchanged for randomness. The fourth game, "HYBR", reverses the change for the *mac* key, to be generated from a random secret again. Finally, in the last game "RAND", the *mac* key is reversed to be generated from the exchanged secret, and only the session key is set to a random value. Figure 4 shows how the five hybrids correspond to the individual lemmata to show indistinguishability of real and random outputs of the TEST query in G_{BRSEC}.

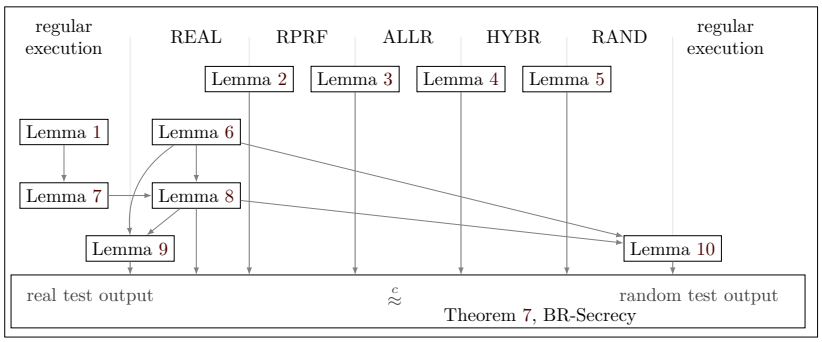

Figure 4: Overview of the lemmas in our proof of BR-Secrecy for the LDACS MAKE protocol. The five middle columns resemble the hybrid games according to the proof of SK-security in [CK02].

We define four different events which may occur during G_{BRSEC} or one of the hybrids: An ABORT event happens when the $\hat{S}$ simulator aborts, *i.e.*, it guessed the test session or its partner incorrectly or the adversary manipulated the communication between the two local sessions. An AFFIRM event happens when the adversary sets b_{guess} to 1. Since the adversary acts as a distinguisher, we can express the indistinguishability of the hybrids via a negligible difference in the probabilities of the AFFIRM event. A GUESS event happens when the $\hat{S}$ simulator guesses correctly. A SIG-FORGE event happens when the adversary successfully forges a signature from one of the protocol parties.

As a first step, our proof constructs Lemmas 2 to 5, which show that the probabilities of the four events only differ by a negligible amount between the different variants of the $\hat{S}$ simulator (cf. Appendix A1.1.3). Since we get $\hat{S}_{\text{RPRF}}$ from $\hat{S}_{\text{REAL}}$ and $\hat{S}_{\text{HYBR}}$ from $\hat{S}_{\text{RAND}}$ by replacing the shared secret of the two guessed local sessions with a random value, and since in our case, the shared secret is a *kem* key, we can use the IND-CPA property of the *kem* to show the negligible difference between these simulators. We get $\hat{S}_{\text{ALLR}}$ from $\hat{S}_{\text{RPRF}}$ or from $\hat{S}_{\text{HYBR}}$ by replacing the outputs of the *prf* with random values for the two guessed local sessions. Therefore, the negligible difference between these simulators follows from the *prf* property.

Subsequently, we make a connection between the hybrid games and the G_{BRSEC} game. For this purpose, we first define Lemmas 6 to 8. Lemma 6 states that an ABORT and a GUESS event cannot coincide under an $\hat{S}_{\text{REAL}}$ simulator. Recall that a GUESS event implies

that the simulator guesses the test session and its partner correctly. This especially means that a local session exists which will eventually become the partner of the test session, unless it aborts. In our case, that local session has the same ecid as the test session. The ecid of a local session is composed of the identities of the two parties involved in the current protocol session as well as the public key and the ciphertext for the *kem*, which also are all the values and identities that the adversary could have manipulated via the exchanged messages. Thus, in case of a GUESS event, the simulator guesses the test session and its partner correctly and the adversary does not manipulate the communication between the two, *i.e.* the simulator will not abort.

Lemma 7 shows that each local session which accepts and has an uncorrupted intended partner has the same ecid as some other local session. To ensure identical ecids, either of the two exchanged signatures is sufficient since it covers the public key and the ciphertext for the *kem* as well as the identity of the party which receives the signature. The identity of the signer is implicitly associated to the verification key. The lemma is useful for Lemma 8, which establishes a noticeable lower bound for a GUESS event. The probability of correctly guessing two specific local sessions is trivially noticeable in the size of the set of all local sessions. However, the test session may not even have a partner, so we also need to use Lemma 7 to limit the probability for this case.

Now we are able to show in Lemmas 9 and 10 that the "REAL" hybrid given a GUESS event is equivalent to G_{BRSEC} with real TEST output and that the "RAND" hybrid given a GUESS event is equivalent to G_{BRSEC} with random TEST output, except for a negligible probability. The reasoning is that, by Lemma 6, a GUESS event prevents the simulator from aborting. If the simulator does not abort, then all local sessions behave consistently and thus the adversary cannot distinguish between the simulation of the hybrids and G_{BRSEC}. Finally, we combine all game hops in Lemmas 2 to 5, 9 and 10 to prove the BR-secrecy of the LDACS MAKE protocol. Although we now only consider the hybrids under the precondition that a GUESS event happens, the game hops of Lemmas 2 to 5 are still valid since Lemma 8 proved the probability of a GUESS event to be noticeable.

□

4.2 Consequential Security Properties

Following the work of De Saint Guilhem et al. [dSGFW19], the Theorems 2 to 7 also imply the following security notions:

Corollary 1 (Full Explicit Entity Authentication, cf. Definition 10)**.** *The* FEXENTAUTH *predicate is fulfilled for the air station AS if sig is a secure signature scheme:*

$$\forall \text{ adversaries } \mathcal{A} : \text{Adv}^{G_{\text{FEXENTAUTH}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq n \cdot \varepsilon_{sig}$$

This follows from IENTAUTH ∧ FENTCONF ⇔ FEXENTAUTH [dSGFW19, Prop. 9.3.] and from Theorems 3 and 4.

Corollary 2 (Almost-Full Explicit Entity Authentication, cf. Definition 11)**.** *The* AFEXENTAUTH *predicate is fulfilled for the ground station GS if sig is a secure signature scheme:*

$$\forall \text{ adversaries } \mathcal{A} : \text{Adv}^{G_{\text{AFEXENTAUTH}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq n \cdot \varepsilon_{sig}$$

This follows from IENTAUTH ∧ AFENTCONF ⇔ AFEXENTAUTH, which can be proven in an analogous way to [dSGFW19, Thm. 3.1.], and from Theorems 3 and 5.

Corollary 3 (Key-Match Soundness)**.** *The* KMSOUNDNESS *predicate is fulfilled for both parties if kem, prf and sig are secure:*

$$\begin{aligned}&\forall \text{ adversaries } \mathcal{A} : \text{Adv}^{G_{\text{KMSOUNDNESS}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa)\\ &\leq l^2 \cdot \left(\frac{4(\varepsilon_{kem}+\varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem}+\varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem}+\varepsilon_{prf})\right) + \left(\frac{l^2}{2^{2\kappa}} + l\lambda_{kem}\right) + \frac{1}{2^\kappa}\end{aligned}$$

This follows from Theorems 2 and 7 and from [dSGFW19, Theorem 5.1.], which states that, with session key space $\mathcal{K}_\Pi$, the following holds:

$$\forall \text{ adversaries } \mathcal{A} \; \exists \text{ adversaries } \mathcal{B}_1, \mathcal{B}_2 :$$
$$\mathrm{Adv}^{G_{\text{KMSoundness}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) \leq l^2 \cdot \mathrm{Adv}^{G_{\text{BRSec}}}_{\mathcal{B}_2,\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) + \mathrm{Adv}^{G_{\text{Match}}}_{\mathcal{B}_1,\Pi,\mathcal{I},\mathcal{I}}(1^\kappa) + \frac{1}{|\mathcal{K}_\Pi|}.$$

Corollary 4 (Implicit Key Authentication)**.** *The iKeyAuth predicate is fulfilled for both parties if kem, prf and sig are secure:*

$$\forall \text{ adversaries} \mathcal{A} : \mathrm{Adv}^{G_{\text{iKeyAuth}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa)$$
$$\leq l^2 \cdot \left(\frac{4(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{prf}) \right) + 2 \cdot \left(\frac{l^2}{2^{2\kappa}} + l\lambda_{kem} \right) + \frac{1}{2^\kappa}$$

This follows from iKeyAuth $\Leftarrow$ iEntAuth $\wedge$ Match $\wedge$ KMSoundness [dSGFW19, Prop. 9.2.] and from Theorems 2 and 3 and Corollary 3.

Corollary 5 (Full Explicit Key Authentication, cf. Definition 12)**.** *The fexKeyAuth predicate is fulfilled for the air station AS if kem, prf and sig are secure:*

$$\forall \text{ adversaries} \mathcal{A} : \mathrm{Adv}^{G_{\text{fexKeyAuth}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa)$$
$$\leq l^2 \left(\frac{4(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{prf}) \right) + 2 \left(\frac{l^2}{2^{2\kappa}} + l\lambda_{kem} \right) + \frac{1}{2^\kappa} + n\varepsilon_{sig}$$

This follows from fexKeyAuth $\Leftarrow$ fexEntAuth $\wedge$ Match $\wedge$ KMSoundness [dSGFW19, Prop. 9.2.] and from Theorem 2 and Corollaries 1 and 3.

Corollary 6 (Almost-Full Explicit Key Authentication, cf. Definition 13)**.** *The afexKeyAuth predicate is fulfilled for the ground station GS if kem, prf and sig are secure:*

$$\forall \text{ adversaries} \mathcal{A} : \mathrm{Adv}^{G_{\text{afexKeyAuth}}}_{\mathcal{A},\Pi,\mathcal{I},\mathcal{I}}(1^\kappa)$$
$$\leq l^2 \left(\frac{4(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{prf}) \right) + \frac{2l^2}{2^{2\kappa}} + 3l\lambda_{kem} + \frac{1}{2^\kappa} + n\varepsilon_{sig}$$

This follows from iKeyAuth $\wedge$ afKeyConf $\Leftrightarrow$ afexKeyAuth [dSGFW19, Thm. 3.1.] and from Theorem 6 and Corollary 4.

5 Symbolic Proof

In this section we review the automated proof from the Tamarin implementation of the LDACS MAKE protocol as described in Section 3. We have made our Tamarin source code for the symbolic proof available at https://github.com/mtiepelt/ldacs-make-symbolic-tamarin.

5.1 Transition Model

Figure 5 outlines the flow of the Tamarin code relative to the description of the protocol in Figure 3 as a set of states, rules and protocol messages for each of the agents. Recall that agent A corresponds to the ground station GS and agent B to the air station AS.

The states of the agents are "S_A_i" for $i \in \mathbb{Z}$ for agent A and "S_B_i" for agent B respectively. The rules, corresponding to the transition between the states of the individual agents (here A) within the protocol, are "*init_A*", "*A_1*", "*A_2*", etc., as well as the

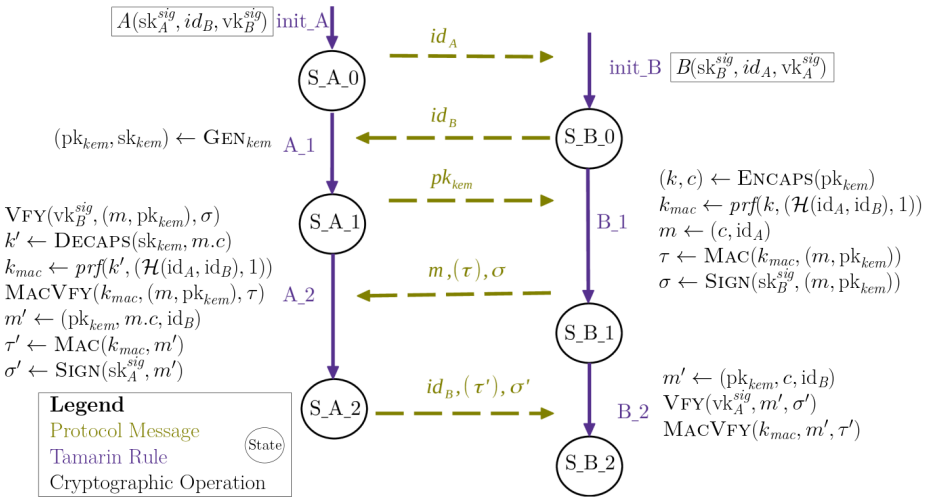

Figure 5: Tamarin rules, states, cryptographic messages and operations for the KEM variant of the protocol. This figure was inspired by [CHSW22b, Fig. 2].

queries accessible to the adversary, for example "Reveal_ltk". Additionally, Figure 5 shows the Protocol messages implied by the rules, as well as the cryptographic messages applied in Figure 3. It may be noted that in the symbolic model the parties first exchange the identifier of the intended partner in plain before engaging in the protocol (in comparison to taking the respective identifier as an input in Figure 3).

Our Tamarin code comprises four implementations of the protocol: First, the expected protocol Figure 3 with the key encapsulation mechanism. Second, the protocol but excluding the generation, transmission and verification of MACs, to verify that all lemmas can be achieved without using the MAC. The third and fourth variant correspond to the protocol using DH instead of a KEM, thus resembling the original KAM-7 [co21, §11.7] for completeness.

For each variant, Tamarin exhaustively explores the possible actions of the agents and the adversary to check if the lemmas *mutual_authentication_A/B*, *session_uniqueness*, *secrecy*, *secrecy_pfs* and *key_consistency_A/B* as described in Section 2.3 are fulfilled. Additionally, the model includes a lemma *session_exists* and *two_sessions_exist*, with the first lemma showing the possibility for the Tamarin model to terminate and the second even enhancing the attacker's abilities by allowing them to reuse cryptographic primitives from the previous protocol run.

5.2 Results

Table 1 shows the results of verifying the lemmas with Tamarin: the "Scope[-traces]" correspond to either the existence of a sequence of traces ("Exists"), or the exhaustive search over all ("All") possible combinations of traces that achieve the conditions required to fulfill the lemma. The "Steps" indicate the number of rules (see Figure 5) that had to be explored to validate the result.

Table 1 shows that both lemmas, *session_exists* and *two_sessions_exist*, have a trace, guaranteeing the completion of a protocol run and that the lemmas hold even if the adversary re-uses values from previous runs. Further, for all combinations of setups (*i.e.*, with Diffie-Hellman or KEM, with or without MAC) the lemmas are satisfied, thus the protocol fulfills the security claims in the symbolic model. Particularly, the lemmas hold even in the setting with the HMAC removed from the protocol, thus the automated proof confirms the findings of Section 4, verifying that the message authentication code is not

Table 1: Tamarin results for LDACS security notions. Modeled in-/excluding a MAC and using DHKE/KEM as key exchange.

Lemma	Scope [-traces]	#Steps w HMAC DH	KEM	#Steps w/o HMAC DH	KEM
session_exists	Exists	24 ✔	25 ✔	23 ✔	23 ✔
two_sessions_exist	Exists	46 ✔	48 ✔	44 ✔	44 ✔
mutual_authentication_A/B	All	50 ✔	54 ✔	50 ✔	54 ✔
session_uniqueness_A/B	All	32 ✔	32 ✔	32 ✔	32 ✔
secrecy	All	32 ✔	28 ✔	24 ✔	26 ✔
secrecy_pfs	All	32 ✔	28 ✔	24 ✔	26 ✔
key_consistency	All	16 ✔	16 ✔	16 ✔	16 ✔

required to provide security for LDACS MAKE in the setting of weak corruption.

6 Bibliography

References

[Aer21] Aeronautical Radio, Incorporated (ARINC). Internet Protocol Suite (IPS) for Aeronautical Safety Services Part 1 Airborne IPS System Technical Requirements. https://standards.globalspec.com/std/14391274/858p1, accessed July 31, 2022, 06 2021.

[BBF+19] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 206–226, Chongqing, China, May 8–10, 2019. Springer, Heidelberg, Germany. doi:10.1007/978-3-030-25510-7_12.

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany. doi:10.1007/11818175_36.

[BFWW11] Christina Brzuska, Marc Fischlin, Bogdan Warinschi, and Stephen C. Williams. Composability of Bellare-Rogaway key exchange protocols. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011: 18th Conference on Computer and Communications Security*, pages 51–62, Chicago, Illinois, USA, October 17–21, 2011. ACM Press. doi:10.1145/2046707.2046716.

[BMS20] Colin Boyd, Anish Mathuria, and Douglas Stebila. *Protocols for Authentication and Key Establishment, Second Edition*. Information Security and Cryptography. Springer, 2020. doi:10.1007/978-3-662-58146-9.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances*

in Cryptology – EUROCRYPT 2000, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer, Heidelberg, Germany. doi:10.1007/3-540-45539-6_11.

[CHSW22a] Sofía Celi, Jonathan Hoyland, Douglas Stebila, and Thom Wiggers. A Tale of Two Models: Formal Verification of KEMTLS via Tamarin. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian D. Jensen, and Weizhi Meng, editors, *Computer Security – ESORICS 2022*, pages 63–83, Cham, 2022. Springer Nature Switzerland. doi:10.1007/978-3-031-17143-7_4.

[CHSW22b] Sofía Celi, Jonathan Hoyland, Douglas Stebila, and Thom Wiggers. A tale of two models: formal verification of KEMTLS via tamarin. Cryptology ePrint Archive, Report 2022/1111, 2022. https://eprint.iacr.org/2022/1111.

[CK02] Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 143–161, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. https://eprint.iacr.org/2002/120/. doi:10.1007/3-540-45708-9_10.

[co21] ISO copyright office. ISO/IEC 11779-3, 2021. URL: https://www.iso.org/standard/82709.html.

[Cre11] Cas J. F. Cremers. Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2. In Vijay Atluri and Claudia Díaz, editors, *ESORICS 2011: 16th European Symposium on Research in Computer Security*, volume 6879 of *Lecture Notes in Computer Science*, pages 315–334, Leuven, Belgium, September 12–14, 2011. Springer, Heidelberg, Germany. doi:10.1007/978-3-642-23822-2_18.

[DFW20] Cyprien Delpech de Saint Guilhem, Marc Fischlin, and Bogdan Warinschi. Authentication in key-exchange: Definitions, relations and composition. In Limin Jia and Ralf Küsters, editors, *CSF 2020: IEEE 33rd Computer Security Foundations Symposium*, pages 288–303, Boston, MA, USA, June 22–26, 2020. IEEE Computer Society Press. doi:10.1109/CSF49147.2020.00028.

[DM17] Jason A Donenfeld and Kevin Milner. Formal verification of the WireGuard protocol. *Technical Report, Tech. Rep.*, 2017. URL: https://www.wireguard.com/papers/wireguard-formal-verification.pdf.

[dSGFW19] Cyprien Delpech de Saint Guilhem, Marc Fischlin, and Bogdan Warinschi. Authentication in Key-Exchange: Definitions, Relations and Composition. Cryptology ePrint Archive, Paper 2019/1203, 2019. https://eprint.iacr.org/2019/1203. URL: https://eprint.iacr.org/2019/1203, doi:10.1109/CSF49147.2020.00028.

[DY83] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. doi:10.1109/TIT.1983.1056650.

[EUR23] EUROCONTROL. SatCOM, 2023. URL: https://www.eurocontrol.int/system/satellite-communications-datalink.

[FKMS20] S. Fluhrer, P. Kampanakis, D. McGrew, and V. Smyslov. Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security. RFC 8784 (Proposed Standard), June 2020. URL: https://www.rfc-editor.org/rfc/rfc8784.txt, doi:10.17487/RFC8784.

[fST15] National Institute for Standards and Technology. Secure Hash Standard. FIPS 180-4, 2015. URL: https://csrc.nist.gov/pubs/fips/180-4/upd1/final.

[fST22] National Institute for Standards and Technology. Post Quantum Cryptography, 2022. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[GGCG+21] Stefan-Lukas Gazdag, Sophia Grundner-Culemann, Tobias Guggemos, Tobias Heider, and Daniel Loebenberger. A Formal Analysis of IKEv2's Post-Quantum Extension. In *Proceedings of the 37th Annual Computer Security Applications Conference*, ACSAC '21, page 91–105, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3485832.3485885.

[Int21] International Civil Aviation Organization (ICAO). ICAO - ANNEX 10 VOL III AMD 91 Aeronautical Telecommunications Volume III - Communications Systems (Part I - Digital Data Communication Systems; Part II - Voice Communication Systems) . https://standards.globalspec.com/std/14383365/ANNEX%2010%20VOL%20III%20AMD%2091, accessed July 31, 2022, 03 2021.

[JU23a] SESAR JU. LDACS A/G Specification, Edition 01.01.00, Template Edition 02.00.05, Edition date 25.04.2023. Technical report, SESAR JU, 2023. Earlier version can be found here https://www.ldacs.com/wp-content/uploads/2023/03/SESAR2020_PJ14-W2-60_TRL6_D3_1_230_3rd_LDACS_AG_Specification_v1.0.0.pdf. URL: https://www.ldacs.com/publications-and-links/#post-Specifications.

[JU)23b] Single European Sky ATM Research Joint Undertaking (SESAR 3 JU). Single European Sky ATM Research, 2023. [Online; accessed 29 December 2023]. URL: https://www.sesarju.eu/.

[KHN+14a] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Internet Key Exchange Protocol Version 2. https://datatracker.ietf.org/doc/html/rfc7296, accessed Oct 01, 2023, 2014. Additional authors: Microsoft, VPN Consortium, Check Point, Independent, INSIDE SECURE. doi:10.17487/RFC7296.

[KHN+14b] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen, and T. Kivinen. Minimal Internet Key Exchange Protocol Version 2. https://www.rfc-editor.org/rfc/rfc7815, accessed Oct 01, 2023, 2014. Additional authors: Microsoft, VPN Consortium, Check Point, Independent, INSIDE SECURE. doi:10.17487/RFC7815.

[Kra03] Hugo Krawczyk. SIGMA: the 'sign-and-mac' approach to authenticated diffie-hellman and its use in the ike-protocols. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 400–425. Springer, 2003. doi:10.1007/978-3-540-45146-4_24.

[Low97] G. Lowe. A Hierarchy of Authentication Specifications. In *Proceedings 10th Computer Security Foundations Workshop*, pages 31–43, Rockport, MA, USA, 1997. doi:10.1109/CSFW.1997.596782.

[MGC22] Nils Mäurer and Sophia Grundner-Culemann. Formal verification of the ldacs make protocol. crypto day matters 34, 2022. doi:10.18420/cdm-2022-34-24.

[MGG+21] Nils Mäurer, Thomas Gräupl, Christoph Gentsch, Tobias Guggemos, Marcel Tiepelt, Corinna Schmitt, and Gabi Dreo Rodosek. A secure cell-attachment procedure of ldacs. In *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 113–122, 2021. doi:10.1109/EuroSPW54576.2021.00019.

[MGS21] Nils Mäurer, Thomas Gräupl, and Corinna Schmitt. Cybersecurity for the l-band digital aeronautical communications system (ldacs). Technical report, German Aerospace Center, 06 2021. doi:10.1049/SBRA545E_ch4.

[MSCB13] S. Meier, B. Schmidt, C. Cremers, and D. Basin. The TAMARIN Prover For The Symbolic Analysis Of Security Protocols. In *25th International Conference on Computer Aided Verification (CAV)*, pages 696–701, Saint Petersburg, Russia, 2013. doi:10.1007/978-3-642-39799-8_48.

[NS78] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM*, 21(12):993–999, dec 1978. doi:10.1145/359657.359659.

[oI23] International Civil Aviation organization (ICAO). CHAPTER 13 L-Band Digital Aeronautical Communications System (LDACS). Technical report, International Civil Aviation organization (ICAO), 2023. URL: https://www.ldacs.com/wp-content/uploads/2023/03/WP06.AppA-DCIWG-6-LDACS_SARPs.pdf.

[Pei14] Chris Peikert. Lattice cryptography for the internet. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*, pages 197–219, Waterloo, Ontario, Canada, October 1–3, 2014. Springer, Heidelberg, Germany. doi:10.1007/978-3-319-11659-4_12.

[Pei15] Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. https://eprint.iacr.org/2015/939.

[PF14] Monica Paolini and Senza Fili. Enabling the Next Generation in Air Traffic Management with AeroMACS. https://files.wimaxforum.org/Document/Download/AeroMACS-Delivering_Next_Generation_Communications_to_the_Airport_Surface, 2014.

[Tea23] The Tamarin Team. Tamarin-Prover Manual, 2023. [Online; accessed 29 December 2023]. URL: https://tamarin-prover.com/manual/master/tex/tamarin-manual.pdf.

[TMM24] Marcel Tiepelt, Christian Martin, and Nils Maeurer. Post-quantum ready key agreement for aviation. *IACR Communications in Cryptology*, 1(1), 2024. doi:10.62056/aebn2isfg.

[WVOW92] Diffie Whitfield, Paul C. Van Oorshot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 1992. doi:10.1007/BF00124891.

Supplementary Material

A1 Computational Security of LDACS MAKE

Proof of Lemma 1, *i.e.*, $\mathbb{P}[\text{SIG-FORGE}] \leq n \cdot \varepsilon_{sig}$

Proof. Let $\mathcal{I}$ be a set of protocol parties, LSID a set of local sessions and $\mathcal{A}$ an adversary that successfully forges one of the signatures in a run of the LDACS MAKE protocol with a non-negligible probability. Then we can construct an equally bounded adversary $\mathcal{B}_{sig}$ from $\mathcal{A}$ which wins the EUF-CMA security game for *sig* with a non-negligible probability:

The EUF-CMA challenger provides a verification key $\text{vk}^* \in \mathcal{VK}_{sig}$ for $\mathcal{B}_{sig}$ and grants $\mathcal{B}_{sig}$ access to a signing oracle $\mathcal{O}_{\text{SIGN}}(\cdot)$. $\mathcal{B}_{sig}$ then samples a party identifier $\text{id}_P \xleftarrow{\$} \mathcal{I}$. It simulates a regular execution of the LDACS MAKE protocol with adversary $\mathcal{A}$, except that the verification key of P is vk^* and P produces signatures via $\mathcal{O}_{\text{SIGN}}(\cdot)$. If at any point $\mathcal{A}$ forges a signature $\sigma' \in \mathcal{S}_{sig}$ by P of a value $x' \in \mathcal{M}_{sig}$ such that the conditions of a SIG-FORGE event are satisfied, then $\mathcal{B}_{sig}$ returns (x', σ') to the EUF-CMA challenger. If $\mathcal{A}$ terminates without successfully forging a signature by P, then $\mathcal{B}_{sig}$ aborts by returning $\perp$.

The view of $\mathcal{A}$ under $\mathcal{B}_{sig}$ is identical to its view in a regular interaction with the LDACS MAKE protocol with n parties. If $\mathcal{A}$ successfully forges a signature for some party, then the probability that $\mathcal{B}_{sig}$ guesses the party correctly is $1/n$. In that case, $\mathcal{B}_{sig}$ returns a valid forgery. This means

$$\frac{1}{n} \cdot \mathbb{P}[\text{SIG-FORGE}] \leq \text{Adv}_{sig}^{\text{EUF-CMA}}(\mathcal{B}_{sig}, \kappa) \leq \varepsilon_{sig} \tag{2}$$

and consequently,

$$\mathbb{P}[\text{SIG-FORGE}] \leq n \cdot \varepsilon_{sig}. \tag{3}$$

□

A1.1 Full Proof of BR-Secrecy

A1.1.1 Simulators

In order to prove BR-secrecy, we need a set of simulators which simulate modified versions of the LDACS MAKE protocol. The proof will contain multiple game hops between different simulators. The first simulator is S, which simulates the LDACS MAKE protocol with no visible modifications.

Definition 15 (Simulator S)**.** The simulator S has the following parameters: a set $\mathcal{I}$ of protocol parties, a set LSID of local sessions of parties from $\mathcal{I}$, the security parameter κ and an adversary $\mathcal{A}$. We define $n := |\mathcal{I}|$ and $l := |\text{LSID}|$. At the beginning of its run, S samples $\ell_{GS}, \ell_{AS} \xleftarrow{\$} \text{LSID}$. S then simulates a run of the BR-secrecy game with the LDACS MAKE protocol against adversary $\mathcal{A}$. Whenever $\mathcal{A}$ sends a query, S updates the state of the parties and answers the query in accordance with the definition of the query in the model. When the simulation terminates, S terminates with output b_{guess}.

Remark 3. Sampling ℓ_{GS} and ℓ_{AS} is only necessary to make sure that a GUESS event (cf. Definition 19) is defined under S.

Table 3 shows where the security loss of Theorem 1 originates from in the proof.

The other type of simulator is $\hat{S}$, which has five different variants. We will use $\hat{S}$ in the proof of implicit key authentication.

Table 2: Relation between our proof of BR-secrecy and the proof of SK-security for the basic SIGMA protocol [CK02].

Our lemmas	SK-security proof [CK02]	Modifications / Comment
Lemma 1		Additional lemma
Lemma 2	Part of [CK02, Lem. 15]	More events, $DDH \rightsquigarrow kem$ (cf. [Pei15]).
Lemma 3	Part of [CK02, Lem. 15]	More events.
Lemma 4	Part of [CK02, Lem. 15]	More events.
Lemma 5	[CK02, Lem. 8 and 10]	More events, $DDH \rightsquigarrow kem$ (cf. [Pei15]).
Lemma 6	[CK02, Lem. 7]	Explicitly presume Guess
Lemma 7	[CK02, Lem. 11]	SameECID instead of Matching sessions.
Lemma 8	[CK02, Lem. 9]	Restated for LDACS MAKE.
Lemma 9	[CK02, Lem. 14]	Restated for LDACS MAKE.
Lemma 10	[CK02, Lem. 13]	Restated for LDACS MAKE.

Table 3: Derivation of security loss of Theorem 1.

Security Notion		
fexKeyAuth (Definition 12)	$\Leftarrow$	fexEntAuth $\wedge$ Match $\wedge$ KMSoundness (Corollary 5)
afexKeyAuth (Definition 13)	$\Leftrightarrow$	iKeyAuth $\wedge$ afKeyConf (Corollary 6)
fexEntAuth (Definition 10)	$\Leftrightarrow$	iEntAuth $\wedge$ fEntConf (Corollary 1)
afexEntAuth (Definition 11)	$\Leftrightarrow$	iEntAuth $\wedge$ afEntConf (Corollary 2)
Match (Definition 5)		Theorem 2
KMSoundness	$\Leftarrow$	BRSec $\wedge$ Match (Corollary 3)
iKeyAuth	$\Leftarrow$	iEntAuth $\wedge$ Match $\wedge$ KMSoundness (Corollary 4)
afKeyConf		Theorem 6
iEntAuth		Theorem 3
fEntConf		Theorem 4
afEntConf		Theorem 5

Definition 16 (Simulator $\hat{S}$)**.** Like S, $\hat{S}$ runs on parameters $\mathcal{I}$, LSID, κ and $\mathcal{A}$. At the beginning of its run, $\hat{S}$ samples $\ell_{GS}, \ell_{AS} \xleftarrow{\$}$ LSID, $(\mathrm{pk}_{kem}, \mathrm{sk}_{kem}) \leftarrow \textsc{Gen}_{kem}(1^\kappa)$ and $(k, c) \leftarrow \textsc{Encaps}(\mathrm{pk})$. It also generates the keys K and k_{mac} as soon as the necessary values for generating them are known. How K and k_{mac} are generated depends on the variant of $\hat{S}$. If at any point during its execution an Abort event as in Definition 18 occurs, $\hat{S}$ terminates with output 0. Like S, $\hat{S}$ simulates a run of the BR-secrecy game against adversary $\mathcal{A}$. If $\mathcal{A}$ performs a query on local sessions which do not include ℓ_{GS} or ℓ_{AS}, $\hat{S}$ updates the state of the respective parties and answers the query in accordance with the definition of the query in the model. For ℓ_{GS} and ℓ_{AS}, $\hat{S}$ acts the same except that the values pk_{kem}, c, K and k_{mac} are replaced by their counterparts sampled by $\hat{S}$. Additionally, the Test query output for ℓ_{GS} and ℓ_{AS} is always K. If the simulation terminates, then $\hat{S}$ terminates with output b_{guess}.

Definition 17 ($\hat{S}$ Variants)**.** The $\hat{S}$ simulator has five variants (corresponding to [CK02, §4.3.2]), which calculate the keys K and k_{mac} for ℓ_{GS} and ℓ_{AS} in multiple ways. Let k be the encapsulated key generated by $\hat{S}$ and let $h := \mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS})$. The five variants

calculate K and k_{mac} in the following ways:

$$
\begin{array}{lllll}
\hat{S}_{\text{REAL}}: & \text{K} := prf(k,(h,0)), & k_{mac} := prf(k,(h,1)) & \\
\hat{S}_{\text{RPRF}}: & \text{K} := prf(k',(h,0)), & k_{mac} := prf(k',(h,1)), & k' \xleftarrow{\$} \mathcal{K}_{prf} \\
\hat{S}_{\text{ALLR}}: & \text{K} \xleftarrow{\$} \mathcal{Y}_{prf}, & k_{mac} \xleftarrow{\$} \mathcal{Y}_{prf} & \\
\hat{S}_{\text{HYBR}}: & \text{K} \xleftarrow{\$} \mathcal{Y}_{prf}, & k_{mac} := prf(k',(h,1)), & k' \xleftarrow{\$} \mathcal{K}_{prf} \\
\hat{S}_{\text{RAND}}: & \text{K} \xleftarrow{\$} \mathcal{Y}_{prf}, & k_{mac} := prf(k,(h,1)) &
\end{array}
$$

A1.1.2 Events

During an execution of the LDACS MAKE protocol or one of the simulators, different events may occur. We will define specific events (extending the work of [CK02, §4.3.2]) which will be important in order to prove the security properties. All of these events can be efficiently detected in an execution of each of the simulators defined above. The first type of events defines when an $\hat{S}$ simulator aborts its execution.

Definition 18 (Abort Event)**.** An Abort event happens in a run of $\hat{S}$ if and only if one of the following conditions is satisfied:

- $\mathcal{A}$ terminates before having activated ℓ_{GS} and ℓ_{AS}.
- ℓ_{GS} is not an initiator.
- ℓ_{AS} is not a responder.
- ℓ_{AS} receives a message before ℓ_{GS} sent its start message.
- ℓ_{AS} receives a value of pk_{kem} different to the respective value sent by ℓ_{GS}.
- ℓ_{GS} receives the response message before ℓ_{AS} was activated.
- ℓ_{GS} receives a value of c different to the respective value sent by ℓ_{AS}.
- $\mathcal{A}$ performs more than one Test queries.
- $\mathcal{A}$ terminates without performing a Test query.
- $\ell_{\text{test}} \notin \{\ell_{GS}, \ell_{AS}\}$.
- $\ell_{\text{test}}.\delta_{\text{ownr}} = \text{corrupt}$ or $\ell_{\text{test}}.\delta_{\text{peer}} = \text{corrupt}$.
- ℓ_{test} aborts.
- $\ell_{GS}.\text{pid} \neq \ell_{AS}.\text{id}$.
- $\ell_{AS}.\text{pid} \neq \ell_{GS}.\text{id}$.

Guess events define when an $\hat{S}$ simulator guesses the test session and its potential partner session correctly.

Definition 19 (Guess Event)**.** A Guess event happens in a run of S or $\hat{S}$ if and only if $\mathcal{A}$ has activated both ℓ_{GS} and ℓ_{AS}, ℓ_{GS} is an initiator, ℓ_{AS} is a responder, $\ell_{GS}.\text{ecid} = \ell_{AS}.\text{ecid}$, $\mathcal{A}$ performs exactly one Test query and $\ell_{\text{test}} \in \{\ell_{GS}, \ell_{AS}\}$.

The second type of events is used to measure the distribution of $\mathcal{A}$'s guess b_{guess} and how it changes between different simulators.

Definition 20 (Affirm Event)**.** An Affirm event happens in a run of $\hat{S}$ if and only if the $\hat{S}$ simulator outputs 1.

SIG-FORGE events cover the cases in which the adversary forges a signature which is exchanged in the protocol. This means that if a SIG-FORGE event does not happen, then the signatures successfully protect the integrity of the signed values.

Definition 21 (SIG-FORGE Event)**.** A SIG-FORGE event happens in a regular execution of the LDACS MAKE protocol against adversary $\mathcal{A}$ or in a run of $\hat{S}$ if and only if a party P, a value $x' \in \mathcal{M}_{sig}$ and a signature $\sigma' \in \mathcal{S}_{sig}$ exist such that $\mathcal{A}$ sends σ' to some local session, $\text{VFY}(\text{vk}_P^{sig}, x', \sigma') = 1$ and neither was P corrupted nor did P sign x' before $\mathcal{A}$ uses σ'.

In order to work with arbitrary subsets of the events we just defined, we create a set of these events.

Definition 22 (Event Set)**.** We define the event set $\mathcal{E} := \{\text{ABORT}, \text{AFFIRM}, \text{GUESS}, \text{SIG-FORGE}\}$. If an event $X \in \mathcal{E}$ or all events of a subset of events $E \subseteq \mathcal{E}$ happen under an $\hat{S}$ variant $\hat{S}_{\text{VAR}}$, we denote this as an X_{VAR} event or an E_{VAR} event, respectively.

A1.1.3 Lemmas

We start the proof with Lemmas 2 to 5, which we will use to deduce probabilities of events under certain $\hat{S}$ variants from the probabilities of the same events under other $\hat{S}$ variants.

Lemma 2. *Similar to [CK02, Lem. 7] with kem instead of DDH as suggested by [Pei14]*

$$\forall \text{ adversaries } \mathcal{A}\ \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{REAL}] - \mathbb{P}[E_{RPRF}]| \leq \varepsilon_{kem}$$

Proof. If we assume that a set $\mathcal{I}$ of protocol parties, a set LSID of local sessions and an adversary $\mathcal{A}$ exist such that the conjunction of all events in E has different probabilities to occur in $\hat{S}_{\text{REAL}}$ and in $\hat{S}_{\text{RPRF}}$, then we can construct an IND-CPA adversary $\mathcal{D}_1$ for *kem* from $\mathcal{A}$:

At the start, $\mathcal{D}_1$ receives $\text{pk}^* \in \mathcal{PK}_{kem}$, $c^* \in \mathcal{C}_{kem}$ and $k^* \in \mathcal{K}_{kem}$ from the IND-CPA challenger. $\mathcal{D}_1$ then simulates $\hat{S}_{\text{REAL}}$ with arguments $\mathcal{I}$, LSID, κ and $\mathcal{A}$ with the modification that for ℓ_{GS} and ℓ_{AS}, the values pk_{kem}, c and k are replaced by pk^*, c^* and k^*, respectively. If all events in E happen in the modified $\hat{S}_{\text{REAL}}$ simulator, then $\mathcal{D}_1$ terminates with output 1. If the modified $\hat{S}_{\text{REAL}}$ simulator terminates without all events in E happening, then $\mathcal{D}_1$ terminates with output 0. $\mathcal{A}$ cannot notice inconsistencies if ℓ_{GS} and ℓ_{AS} have differing views of the values of pk_{kem} and c, since in this case, the modified $\hat{S}_{\text{REAL}}$ simulator aborts. If k^* is a real encapsulated key, then the view of $\mathcal{A}$ under $\mathcal{D}_1$ is identical to its view under $\hat{S}_{\text{REAL}}$. If k^* is a random key, then the view of $\mathcal{A}$ under $\mathcal{D}_1$ is identical to its view under $\hat{S}_{\text{RPRF}}$. This means

$$|\mathbb{P}[E_{\text{REAL}}] - \mathbb{P}[E_{\text{RPRF}}]| \leq \text{Adv}_{kem}^{\text{IND-CPA}}(\mathcal{D}_1, \kappa) \leq \varepsilon_{kem} \tag{4}$$

□

Lemma 3. *Similar to [CK02, Lem. 15]*

$$\forall \text{ adversaries } \mathcal{A}\ \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{RPRF}] - \mathbb{P}[E_{ALLR}]| \leq \varepsilon_{prf}$$

Proof. If we assume that a set $\mathcal{I}$ of protocol parties, a set LSID of local sessions and an adversary $\mathcal{A}$ exist such that the conjunction of all events in E has different probabilities to occur in $\hat{S}_{\text{RPRF}}$ and in $\hat{S}_{\text{ALLR}}$, then we can construct a PRF adversary $\mathcal{D}_2$ for $prf(\cdot,\cdot)$ from $\mathcal{A}$:

At the start, the *prf* challenger grants $\mathcal{D}_2$ access to an oracle $\mathcal{O}_{prf/rf}(\cdot)$ which implements $prf(k^*,\cdot)$ for a uniformly sampled key $k^* \in \mathcal{K}_{prf}$ or which implements a random function $rf\colon \mathcal{X}_{prf} \to \mathcal{Y}_{prf}$. $\mathcal{D}_2$ then simulates a modified version of $\hat{S}_{\text{RPRF}}$ which uses $\mathcal{O}_{prf/rf}(\cdot)$ instead

of $prf(k', \cdot)$. This means that ℓ_{GS} and ℓ_{AS} use session key $\mathrm{K} := \mathcal{O}_{prf/rf}(\mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS}), 0)$ and MAC key $k_{mac} := \mathcal{O}_{prf/rf}(\mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS}), 1)$. If all events in E happen in the modified $\hat{S}_{\mathrm{RPRF}}$ simulator, then $\mathcal{D}_2$ terminates with output 1. If the modified $\hat{S}_{\mathrm{RPRF}}$ simulator terminates without all events in E happening, then $\mathcal{D}_2$ terminates with output 0. If $\mathcal{O}_{prf/rf}(\cdot) = prf(k^*, \cdot)$, then the view of $\mathcal{A}$ under $\mathcal{D}_2$ is identical to its view under $\hat{S}_{\mathrm{RPRF}}$. If $\mathcal{O}_{prf/rf}(\cdot)$ implements $rf(\cdot)$, then the view of $\mathcal{A}$ under $\mathcal{D}_2$ is identical to the view under $\hat{S}_{\mathrm{ALLR}}$. This means

$$|\mathbb{P}[E_{\mathrm{RPRF}}] - \mathbb{P}[E_{\mathrm{ALLR}}]| \leq \mathrm{Adv}^{\mathrm{PRF}}_{prf}(\mathcal{D}_2, \kappa) \leq \varepsilon_{prf}. \tag{5}$$

□

Lemma 4. *Similar to [CK02, Lem. 15]*

$$\forall \text{ adversaries } \mathcal{A}\ \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{ALLR}] - \mathbb{P}[E_{HYBR}]| \leq \varepsilon_{prf}$$

Proof. If we assume that a set $\mathcal{I}$ of protocol parties, a set LSID of local sessions and an adversary $\mathcal{A}$ exist such that the conjunction of all events in E has different probabilities to occur in $\hat{S}_{\mathrm{ALLR}}$ and in $\hat{S}_{\mathrm{HYBR}}$, then we can construct a PRF adversary $\mathcal{D}_3$ for prf from $\mathcal{A}$, similar to how we constructed $\mathcal{D}_2$ in the proof of Lemma 3. The difference is that $\mathcal{D}_3$ uses $\hat{S}_{\mathrm{HYBR}}$ instead of $\hat{S}_{\mathrm{RPRF}}$. This means effectively that the only difference between $\mathcal{D}_3$ and $\mathcal{D}_2$ is that $\mathcal{D}_3$ uniformly and independently samples K instead of generating it via $\mathcal{O}_{prf/rf}(\mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS}), 0)$. Accordingly, if $\mathcal{O}_{prf/rf}(\cdot) = prf(k^*, \cdot)$, then the view of $\mathcal{A}$ under $\mathcal{D}_3$ is identical to the view under $\hat{S}_{\mathrm{HYBR}}$ instead of $\hat{S}_{\mathrm{RPRF}}$. The inequality we can deduce for this lemma is the following:

$$|\mathbb{P}[E_{\mathrm{ALLR}}] - \mathbb{P}[E_{\mathrm{HYBR}}]| \leq \mathrm{Adv}^{\mathrm{PRF}}_{prf}(\mathcal{D}_3, \kappa) \leq \varepsilon_{prf} \tag{6}$$

□

Lemma 5. *Similar to [CK02, Lem. 8,15] with kem instead of DDH as suggested by [Pei14]*

$$\forall \text{ adversaries } \mathcal{A}\ \forall E \subseteq \mathcal{E} : |\mathbb{P}[E_{HYBR}] - \mathbb{P}[E_{RAND}]| \leq \varepsilon_{kem}$$

Proof. If we assume that a set $\mathcal{I}$ of protocol parties, a set LSID of local sessions and an adversary $\mathcal{A}$ exist such that the conjunction of all events in E has different probabilities to occur in $\hat{S}_{\mathrm{HYBR}}$ and in $\hat{S}_{\mathrm{RAND}}$, then we can construct an IND-CPA adversary $\mathcal{D}_4$ for kem from $\mathcal{A}$, similar to how we constructed $\mathcal{D}_1$ in the proof of Lemma 2. The difference is that $\mathcal{D}_4$ uses $\hat{S}_{\mathrm{RAND}}$ instead of $\hat{S}_{\mathrm{REAL}}$. This means effectively that the only difference between $\mathcal{D}_4$ and $\mathcal{D}_1$ is that $\mathcal{D}_4$ samples K independently instead of generating it via $prf(k^*, (\mathcal{H}(\mathrm{id}_{GS}, \mathrm{id}_{AS}), 0))$. Accordingly, depending on whether k^* is real or random, the view of $\mathcal{A}$ under $\mathcal{D}_4$ is identical to the view under $\hat{S}_{\mathrm{RAND}}$ or $\hat{S}_{\mathrm{HYBR}}$ instead of $\hat{S}_{\mathrm{REAL}}$ or $\hat{S}_{\mathrm{RPRF}}$, respectively. The inequality we can deduce for this lemma is the following:

$$|\mathbb{P}[E_{\mathrm{HYBR}}] - \mathbb{P}[E_{\mathrm{RAND}}]| \leq \mathrm{Adv}^{\text{IND-CPA}}_{kem}(\mathcal{D}_4, \kappa) \leq \varepsilon_{kem} \tag{7}$$

□

With the next lemma, we show that $\hat{S}_{\mathrm{REAL}}$ only aborts with a negligible probability if a Guess event happens. We need this because when comparing a regular execution of the LDACS MAKE protocol with $\hat{S}$, we rely on $\hat{S}$ not aborting under certain circumstances.

Lemma 6. *Similar to [CK02, Lem. 7]*

$$\forall \text{ adversaries } \mathcal{A} : \mathbb{P}[Abort_{REAL} \wedge Guess_{REAL}] = 0$$

Proof. Consider a run of $\hat{S}_{\text{REAL}}$. Assume that a GUESS event happens. By the definition of a GUESS event (cf. Definition 19), $\mathcal{A}$ has activated ℓ_{GS} and ℓ_{AS}, ℓ_{GS} is an initiator, ℓ_{AS} a responder, $\ell_{GS}.\text{ecid} = \ell_{AS}.\text{ecid}$, $\mathcal{A}$ performs exactly one TEST query and $\ell_{\text{test}} \in \{\ell_{GS}, \ell_{AS}\}$. Consider a fixed point in time before which an ABORT event has not happened. $\hat{S}_{\text{REAL}}$ behaves identically to a regular interaction with the LDACS MAKE protocol where the response to the TEST query is the real session key. We can assume without loss of generality that after the next adversary query, $\ell_{\text{test}}.\delta_{\text{ownr}} = \ell_{\text{test}}.\delta_{\text{peer}} = \text{honest} \wedge \ell_{\text{test}}.\delta_{\text{sess}} = \text{fresh} \wedge (\forall \ell' \in \text{LSID}, \ell'.\text{sid} = \ell_{\text{test}}.\text{sid} : \ell'.\delta_{\text{sess}} = \text{fresh}) \wedge (\ell_{\text{test}} \text{ accepts})$ will still hold, since otherwise $\mathcal{A}$ would lose the BR-secrecy game in a regular interaction with the LDACS MAKE protocol. Since $\ell_{GS}.\text{ecid} = \ell_{AS}.\text{ecid}$, it must be the case that either $\ell_{GS}.\text{sid} = \ell_{AS}.\text{sid}$ or only ℓ_{test} accepts and the other local session does not. In both cases, $\mathcal{A}$ neither reveals ℓ_{GS} nor ℓ_{AS}. That both ℓ_{GS} and ℓ_{AS} have the same ECID also means that they have the same view of the values of id_{GS}, id_{AS}, pk_{kem} and c.

The above points mean that, if a GUESS event has already happened, the next adversary query cannot trigger an ABORT event. On the other hand, when an ABORT event happens, $\hat{S}_{\text{REAL}}$ aborts its execution, meaning that a GUESS event cannot happen after an ABORT event. Thus, both an ABORT and a GUESS event cannot happen in the same execution of $\hat{S}_{\text{REAL}}$. □

Lemma 7. *Similar to [CK02, Lem. 11]*

$$\forall \textit{ adversaries } \mathcal{A}\ \forall \ell \in \text{LSID}\ \exists \ell' \in \text{LSID} :$$
$$(\ell \textit{ accepts } \wedge \ell.\delta_{\text{peer}} = \text{honest} \Rightarrow \mathbb{P}[\neg \textsc{SameECID}(\ell, \ell')] \leq n \cdot \varepsilon_{sig})$$

Proof. Let $\ell \in \text{LSID}$ be any local session with $\ell.\text{accept} = \text{true}$ and $\ell.\delta_{\text{peer}} = \text{honest}$. Since ℓ accepts, it has received a valid signature of pk_{kem}, c and $\ell.\text{id}$ by its intended partner. If the intended partner did not create the signature, then $\mathcal{A}$ forged it. By Lemma 1, the probability for this event is at most $n \cdot \varepsilon_{sig}$. If the intended partner created the signature, then a local session $\ell' \in \text{LSID}$ exists which has the same view of the values of id_{GS}, id_{AS}, pk_{kem} and c. In this case, $\ell'.\text{ecid} = \ell.\text{ecid}$. In total, the probability that no local session $\ell' \in \text{LSID}$ exists with the same ECID is at most $n \cdot \varepsilon_{sig}$. □

In the following lemma, we establish a noticeable lower bound for the probability of a GUESS event. Because of this, we know that negligible probabilities do not become non-negligible when we introduce the condition that a GUESS event occurred.

Lemma 8. *Similar to [CK02, Lem. 9]*

$$\forall \textit{ adversaries } \mathcal{A} : \mathbb{P}[\textsc{Guess}_{\textsc{real}}] \geq \frac{1}{l^2} - n\varepsilon_{sig}$$

Proof. We assume without loss of generality that in a regular interaction with the LDACS MAKE protocol, $\mathcal{A}$ chooses exactly one test session ℓ_{test} and that $\mathcal{A}$ does not corrupt ℓ_{test} or its intended partner before ℓ_{test} accepts. This also applies to a simulation of $\mathcal{A}$ under simulator S. By Lemma 7, a local session $\ell' \in \text{LSID}$ exists with $\ell'.\text{ecid} = \ell_{\text{test}}.\text{ecid}$ with a probability of at least $1 - n\varepsilon_{sig}$. If such a local session exists, then the probability that under simulator S, ℓ_{GS} is an initiator, ℓ_{AS} is a responder and $\ell_{\text{test}} \in \{\ell_{GS}, \ell_{AS}\}$, is at least $1/l^2$. In total, the probability of a GUESS event under S is at least

$$\frac{1}{l^2} - n\varepsilon_{sig}.$$

Let $\hat{S}'_{\text{REAL}}$ be a simulator that exactly acts like $\hat{S}_{\text{REAL}}$ except that it never aborts. Since $\mathcal{A}$ has to choose ℓ_{test} before it knows the response to the TEST query, $\mathcal{A}$'s choice of ℓ_{test} has the same distribution under S and under $\hat{S}'_{\text{REAL}}$. Therefore, a GUESS event has the same

probability under S and under $\hat{S}'_{\text{REAL}}$. By Lemma 6 it is not possible that both an ABORT event and a GUESS event happen under $\hat{S}_{\text{REAL}}$. This also means that the probabilities of a GUESS event under $\hat{S}'_{\text{REAL}}$ and under $\hat{S}_{\text{REAL}}$ are identical. We can now deduce:

$$\begin{aligned}
&\mathbb{P}\left[\text{GUESS}_{\text{REAL}}\right] \\
&= \mathbb{P}\left[\text{GUESS under } \hat{S}'_{\text{REAL}}\right] && (8)\\
&= \mathbb{P}\left[\text{GUESS under } S\right] && (9)\\
&\geq \frac{1}{l^2} - n\varepsilon_{sig} && (10)
\end{aligned}$$

□

With the last two lemmas, we show the connection between $\hat{S}$ and a regular execution of the LDACS MAKE protocol. The first of the two lemmas states that $\hat{S}_{\text{REAL}}$ with a GUESS event is indistinguishable from a regular interaction with a real session key as a response to the TEST query.

Lemma 9. *Similar to [CK02, Lem. 14]*

$$\forall \text{ adversaries } \mathcal{A} : |P_{REAL}(\mathcal{A}) - \mathbb{P}\left[AFFIRM_{REAL}|GUESS_{REAL}\right]| = 0$$

Proof. By Lemmas 6 and 8, we know that

$$\begin{aligned}
\mathbb{P}\left[\text{ABORT}_{\text{REAL}} \wedge \text{GUESS}_{\text{REAL}}\right] &= 0\\
\mathbb{P}\left[\text{GUESS}_{\text{REAL}}\right] &\geq \frac{1}{l^2} - n\varepsilon_{sig}.
\end{aligned}$$

The view of $\mathcal{A}$ under $\hat{S}_{\text{REAL}}$ given a GUESS event is the same as in a regular interaction with the LDACS MAKE protocol with real TEST query output unless an ABORT event happens. This means that

$$\begin{aligned}
&|P_{\text{REAL}}(\mathcal{A}) - \mathbb{P}\left[\text{AFFIRM}_{\text{REAL}}|\text{GUESS}_{\text{REAL}}\right]| \\
&\leq \mathbb{P}\left[\text{ABORT}_{\text{REAL}}|\text{GUESS}_{\text{REAL}}\right] && (11)\\
&= \frac{\mathbb{P}\left[\text{ABORT}_{\text{REAL}} \wedge \text{GUESS}_{\text{REAL}}\right]}{\mathbb{P}\left[\text{GUESS}_{\text{REAL}}\right]} && (12)\\
&\leq \frac{0}{\frac{1}{l^2} - n\varepsilon_{sig}} && (13)\\
&= 0. && (14)
\end{aligned}$$

The view of $\mathcal{A}$ under $\hat{S}_{\text{REAL}}$ given a GUESS event is the same as in a regular interaction with the LDACS MAKE protocol with real TEST query output unless an ABORT event happens. This means that since $\mathbb{P}\left[\text{ABORT}_{\text{REAL}}|\text{GUESS}_{\text{REAL}}\right]$ is equal to 0, so is the difference between $P_{\text{REAL}}(\mathcal{A})$ and $\mathbb{P}\left[\text{AFFIRM}_{\text{REAL}}|\text{GUESS}_{\text{REAL}}\right]$. □

The last lemma states that $\hat{S}_{\text{RAND}}$ with a GUESS event is indistinguishable from a regular interaction with a random value as a response to the TEST query.

Lemma 10. *Similar to [CK02, Lem. 13] The following holds true:*

$$\begin{aligned}
&\forall \text{ adversaries } \mathcal{A} : \\
&|P_{RAND}(\mathcal{A}) - \mathbb{P}\left[AFFIRM_{RAND}|GUESS_{RAND}\right]| \leq \frac{2(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}}. \qquad (15)
\end{aligned}$$

Proof. Analogous to Lemma 9, using Lemmas 2 to 5 to argue that

$$\mathbb{P}\left[\textsc{Abort}_{\textsc{rand}} \wedge \textsc{Guess}_{\textsc{rand}}\right] \leq 2(\varepsilon_{kem} + \varepsilon_{prf}) \tag{16}$$

$$\mathbb{P}\left[\textsc{Guess}_{\textsc{rand}}\right] \geq \frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig} \tag{17}$$

follows from

$$\mathbb{P}\left[\textsc{Abort}_{\textsc{real}} \wedge \textsc{Guess}_{\textsc{real}}\right] = 0$$

$$\mathbb{P}\left[\textsc{Guess}_{\textsc{real}}\right] \geq \frac{1}{l^2} - n\varepsilon_{sig}.$$

□

Finalizing the Proof of BR-Secrecy, Theorem 7 Finally, we combine all lemmas to show that in a regular execution of the LDACS MAKE, the key of a session is indistinguishable from an independently sampled random value.

Proof. By Lemmas 2 to 5,

$$\left|\mathbb{P}\left[\textsc{Affirm}_{\textsc{real}} \wedge \textsc{Guess}_{\textsc{real}}\right] - \mathbb{P}\left[\textsc{Affirm}_{\textsc{rand}} \wedge \textsc{Guess}_{\textsc{rand}}\right]\right| \leq 2(\varepsilon_{kem} + \varepsilon_{prf})$$

$$\left|\mathbb{P}\left[\textsc{Guess}_{\textsc{real}}\right] - \mathbb{P}\left[\textsc{Guess}_{\textsc{rand}}\right]\right| \leq 2(\varepsilon_{kem} + \varepsilon_{prf}).$$

By Lemma 8,

$$\mathbb{P}\left[\textsc{Guess}_{\textsc{real}}\right] \geq \frac{1}{l^2} - n\varepsilon_{sig}.$$

By Lemma 9,

$$\left|P_{\textsc{real}}(\mathcal{A}) - \mathbb{P}\left[\textsc{Affirm}_{\textsc{real}} | \textsc{Guess}_{\textsc{real}}\right]\right| = 0.$$

By Lemma 10,

$$\left|P_{\textsc{rand}}(\mathcal{A}) - \mathbb{P}\left[\textsc{Affirm}_{\textsc{rand}} | \textsc{Guess}_{\textsc{rand}}\right]\right| \leq \frac{2(\varepsilon_{kem} + 2\varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}}.$$

Let

$$\begin{aligned}\nu := &\left|P_{\textsc{real}}(\mathcal{A}) - \mathbb{P}\left[\textsc{Affirm}_{\textsc{real}} | \textsc{Guess}_{\textsc{real}}\right]\right| \\ &+ \left|P_{\textsc{rand}}(\mathcal{A}) - \mathbb{P}\left[\textsc{Affirm}_{\textsc{rand}} | \textsc{Guess}_{\textsc{rand}}\right]\right|.\end{aligned}$$

We can now deduce:

$$
\begin{aligned}
&|P_{\text{REAL}}(\mathcal{A}) - P_{\text{RAND}}(\mathcal{A})| \\
&= \big|\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}}|\textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}}|\textsc{Guess}_{\text{RAND}}\right] \\
&\quad + (P_{\text{REAL}}(\mathcal{A}) - \mathbb{P}\left[\textsc{Affirm}_{\text{REAL}}|\textsc{Guess}_{\text{REAL}}\right]) \\
&\quad - (P_{\text{RAND}}(\mathcal{A}) - \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}}|\textsc{Guess}_{\text{RAND}}\right])\big| && (18)\\
&\leq |\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}}|\textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}}|\textsc{Guess}_{\text{RAND}}\right]| + \nu && (19)\\
&= \left|\frac{\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}} \wedge \textsc{Guess}_{\text{REAL}}\right]}{\mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right]} - \frac{\mathbb{P}\left[\textsc{Affirm}_{\text{RAND}} \wedge \textsc{Guess}_{\text{RAND}}\right]}{\mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]}\right| + \nu && (20)\\
&= \mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right]^{-1} \cdot \Big|\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}} \wedge \textsc{Guess}_{\text{REAL}}\right] \\
&\quad - \frac{\mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right]}{\mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]} \cdot \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}} \wedge \textsc{Guess}_{\text{RAND}}\right]\Big| + \nu && (21)\\
&= \mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right]^{-1} \cdot \Big|\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}} \wedge \textsc{Guess}_{\text{REAL}}\right] \\
&\quad - \left(1 + \frac{\mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]}{\mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]}\right) \cdot \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}} \wedge \textsc{Guess}_{\text{RAND}}\right]\Big| + \nu && (22)\\
&\leq \mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right]^{-1} \cdot |\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}} \wedge \textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}} \wedge \textsc{Guess}_{\text{RAND}}\right]| \\
&\quad + \frac{|\mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]|}{\mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]} \cdot \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}} \wedge \textsc{Guess}_{\text{RAND}}\right] + \nu && (23)\\
&\leq \mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right]^{-1} \cdot |\mathbb{P}\left[\textsc{Affirm}_{\text{REAL}} \wedge \textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Affirm}_{\text{RAND}} \wedge \textsc{Guess}_{\text{RAND}}\right]| \\
&\quad + |\mathbb{P}\left[\textsc{Guess}_{\text{REAL}}\right] - \mathbb{P}\left[\textsc{Guess}_{\text{RAND}}\right]| + \nu && (24)\\
&\leq \frac{1}{\frac{1}{l^2} - n\varepsilon_{sig}} \cdot 2(\varepsilon_{kem} + \varepsilon_{prf}) \\
&\quad + 2(\varepsilon_{kem} + \varepsilon_{prf}) + \left(0 + \frac{2(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}}\right) && (25)\\
&\leq \frac{4(\varepsilon_{kem} + \varepsilon_{prf})}{\frac{1}{l^2} - 2(\varepsilon_{kem} + \varepsilon_{prf}) - n\varepsilon_{sig}} + 2(\varepsilon_{kem} + \varepsilon_{prf}) && (26)
\end{aligned}
$$

□

A1.2 Application of proof to IKEv2

If we want to prove the same properties for IKEv2 as we do for the LDACS protocol in Section 4, none of the differences outlined in Section 3.2 affect the proof except for the fact that the parties in IKEv2 do not sign the ID or the DH exponential of the respective other party. To account for the differences, we can choose the ECIDs and SIDs for IKEv2 to contain the two nonces, but not the IDs of the two parties. Since the IDs of the protocol parties are not part of the SIDs, implicit entity authentication (Definition 6) is no longer trivially fulfilled. Instead, it can be argued that for each local session, the received signature confirms that the received nonce was chosen by the intended partner while the probability that the own nonce was coincidentally chosen by another party is negligible.

The proof of almost-full key confirmation (Definition 9) is also affected since the signatures do not sign the DH exponential of the respective other party. A single signature therefore does not contain enough explicit information to unambiguously specify the session key. However, the IKE_AUTH messages use keys derived in the same manner as the session key, so one can argue that the probability of successfully forging an IKE_AUTH message without knowing the session key is negligible.

When proving BR-secrecy for IKEv2 as in Theorem 7, the simulator $\hat{S}$ should not have the two abort conditions $\ell_{GS}.\text{pid} \neq \ell_{AS}.\text{id}$ and $\ell_{AS}.\text{pid} \neq \ell_{GS}.\text{id}$, since the protocol cannot guarantee that neither of the two conditions occur during the BR-secrecy game. $\hat{S}$ should have an abort condition $\ell_{\text{test}}.\text{pid} \notin \{\ell_{GS}.\text{id}, \ell_{AS}.\text{id}\}$ instead of the two conditions previously mentioned. While this makes an abort of $\hat{S}$ easier to avoid, the modification of ECID still makes it possible that an Abort and a Guess event coincide. Lemma 6 therefore should use the signatures to at least show that the probability for this coincidence is negligible.

A2 Alternative LDACS MAKE Protocol

Figure 6 shows an alternative LDACS MAKE protocol, where the two parties additionally use nonces N_{AS} and N_{GS} to derive their *mac* keys as well as the session keys $\mathrm{K}_{GS}, \mathrm{K}_{AS}$. The alternative protocol achieves the same security as outlined in Theorem 8, which is almost identical to Theorem 1, except that the *mac* is now required to prove security. The reason for this is, informally, that the shared key is derived from the nonces, but the air station AS only provides a signature on their own nonce (and vice versa). As such, the ground station GS has no assurance that the AS derives the same key, as they cannot verify, that the AS has knowledge of the nonce N_{GS}. Since both nonces are used to generate the *mac* key via the *prf*, verifying the message authentication key provides the required assurance, and thus the same security guarantees apply.

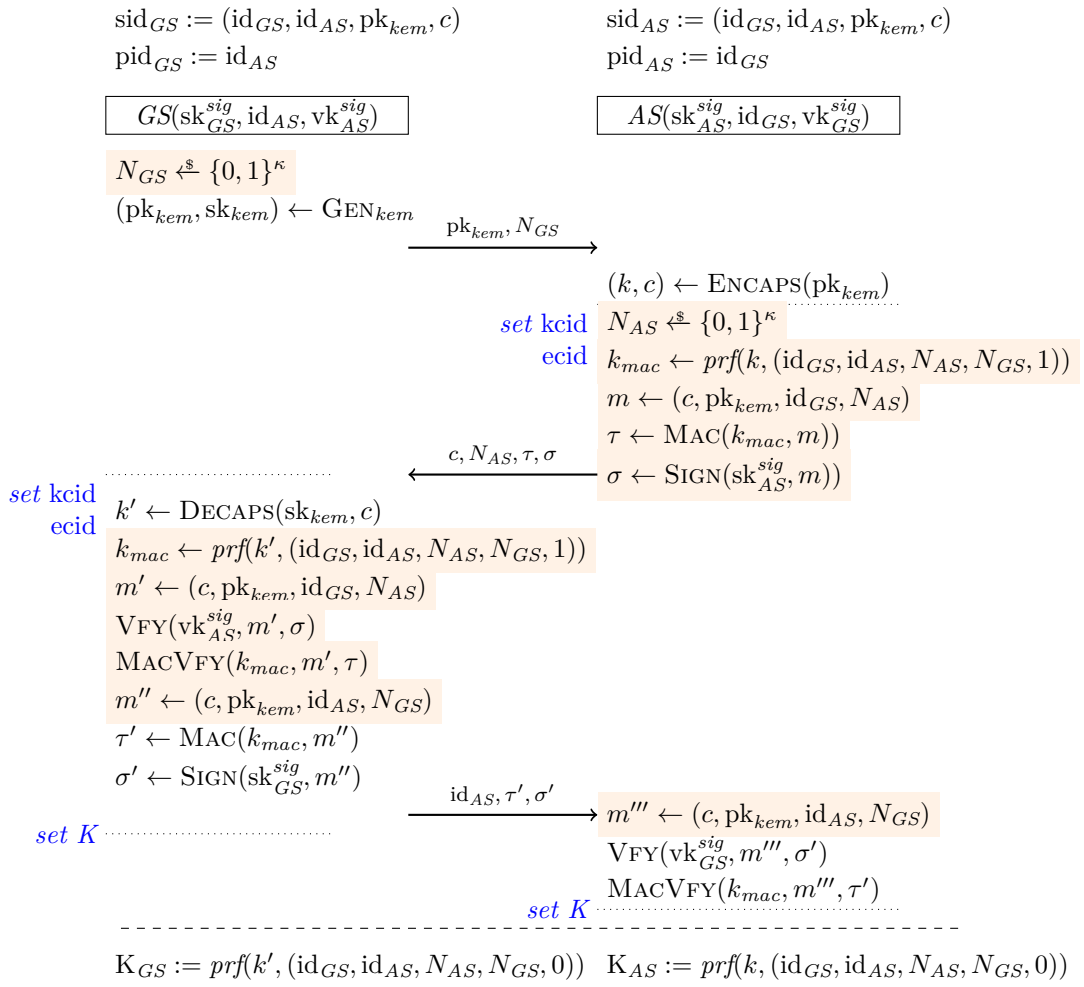

Figure 6: Simplified, alternative key agreement protocol between the ground station GS and the air station AS. If any of the verification $\textsc{Vfy}$ or $\textsc{MacVfy}$ fail, the respective party aborts. The dotted lines with the blue identifiers are relevant for the computational proof in Section 4. Note that this alternative protocol version (compared to the one presented in Figure 3) is currently under investigation by the *German Aerospace Center*. The modification are highlighted in orange shade.

Theorem 8 (Informal)**.** *Let kem be an IND-CPA secure key encapsulation mechanism, let sig be an EUF-CMA secure signature scheme, let prf be an ε_{prf} secure pseudo-random function and let mac be an sEUF-CMA secure message authentication code. Then the alternative LDACS MAKE protocol (cf. Figure 6) provides (almost) full explicit entity authentication and (almost) full explicit key authentication, which implicitly includes BR-secrecy.*

Proof Sketch.(**Informal**)
Define $\text{sid} = \text{kcid} = \text{ecid} = (\text{id}_{GS}, \text{id}_{AS}, \text{pk}_{kem}, c, N_{GS}, N_{AS})$, then the proofs for the properties using Match and iEntAuth predicate carry over with no other modifications. The proofs showing security relative to the BRSec, fEntConf, afEntConf and afKeyConf predicate require adaption to match the modification in the alternative LDACS MAKE protocol. More specifically, the proof now uses a reduction to the sEUF-CMA security of the *mac*. Correspondingly, the deduction of fKeyConf from the Match, iEntAuth predicate as well as from the fEntConf and BRSec remain unchanged, if the security relative to the latter holds.

(Almost) Full entity confirmation and almost-full key confirmation We define three new simulators which are similar to $\hat{S}_{\text{REAL}}$, $\hat{S}_{\text{RPRF}}$ and $\hat{S}_{\text{ALLR}}$, respectively, except that they are not tailored to the BR-secrecy game, meaning that neither do they modify the behavior of the Test query nor do they have any abort conditions that depend on the adversary's usage of the Test query. Additionally, the new simulator which is similar to $\hat{S}_{\text{ALLR}}$, replaces *prf* with a random function instead of sampling the keys directly. Note that this modification still results in random *mac* keys k_{mac}. Therefore, we can use the sEUF-CMA security of *mac* to argue that the adversary is unable to forge a *mac* tag unless it knows the *mac* key. However, we the adversary does not know the random function, they have only negligible chance of guesses the correct key for the two *special* local sessions.

Remember that the input to the *prf* computing the *mac* key consists of the ecid, the secret from the key encapsulation mechanism k as well as the *kem* ciphertext. Therefore, if one of the two special local sessions successfully validates a *mac* tag and the *kem* values have not been manipulated, then both have the same ecid. One can show via a set of game hops using the three simulators that this also applies to a regular execution of the protocol.

Finally, we also consider the signatures which assure the respective receiving party of the authenticity of the *kem* values as long as the sending party is not corrupted. The resulting guaranteed authenticity of the entire ecid, and therefore also guaranteed authenticity of the sid and the kcid, can now serve as a proof for full entity confirmation, almost-full entity confirmation and almost-full key confirmation.

BR-Secrecy The above procedure can also be utilized to show that a variant of Lemma 7 of the BR-secrecy proof holds, except with slightly modified bound on the adversarial advantage. The only remaining modifications of the BR-secrecy proof are that the simulator $\hat{S}$ generates the keys for the two guessed local sessions in the same way as the alternative LDACS MAKE protocol and that $\hat{S}$ also aborts if any of the nonces exchanged between the two get manipulated. Note that the definition of the ecids can be modified to avoid the additional abort condition in case the ecids of the two guessed local sessions are identical. Otherwise, the proof does not require modification.