University of Sheffield

Formalizing, Analyzing and Improving Aerospace Security Protocols

Bhagya Wimalasiri

Supervisors:
Dr. Benjamin Dowling
Prof. John A Clark

Submitted in fulfillment of the requirement

for the degree of Doctor of Philosophy

in the

School of Computer Science

July 31, 2025

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. Work submitted for this research degree at the University of Sheffield has not formed part of any other degree either at the University of Sheffield or at another establishment. All sources are acknowledged as references. Certain parts of the material presented within this thesis have appeared in published or submitted papers elsewhere. Specifically, these are:

Alnashwan, A., Dowling, B., **Wimalasiri, B.** (2025). Path Privacy and Handovers: Preventing Insider Traceability Attacks During Secure Handovers. 38th IEEE Computer Security Foundations Symposium (CSF).

Dowling, B., Hale, B., Tian, X., **Wimalasiri, B.** (2024). Cryptography is Rocket Science: Analysis of BPSec. IACR Communications in Cryptology, 1(4).

Dowling, B., **Wimalasiri, B.** (2024). Quantum-Secure Hybrid Communication for Aviation Infrastructures. IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2024.3483448.

Name: Bhagya Wimalasiri

Signature:

Date: 31.07.2025

Abstract

Secure networking protocols for domains such as the internet and the internet-of-things are regularly scrutinized and discussed by the research community. This status quo can be attributed to their pervasiveness in the modern technological landscape and the heavy dependence on the security of these protocols to reliably fulfill myriad everyday tasks. In contrast, the security of aerospace protocols has not received a proportionate level of investigation despite the fact that an attack on these protocols may result in catastrophic consequences. The reason for this can be twofold; the digitization of avionic communication is an experimental and ongoing process with legacy analogue components still in use; and most non-terrestrial communication protocols are closed-source which limits their public scrutiny. This work endeavors to address this gap in three parallels; we propose a formalism that captures the security of the integral handover phase of aviation communication; we analyze the security of non-terrestrial protocol Bundle Protocol Security (BPSec) and propose security improvements; finally, we propose quantum-secure protocol constructions for aviation communication, supported by empirical results of their performance on resource-constrained devices.

We begin by introducing a universal formalized framework that captures handover schemes as a unique primitive. Moreover, we construct security experiments that capture distinct security properties for handovers, and propose a generic protocol construction that captures all identified security properties. We then shift our focus to the analysis of the non-terrestrial BPSec protocol, which we analyze as a flexible secure channel and propose a stronger construction with additional integrity guarantees. Finally, we return to the aviation domain and propose quantum-secure protocol constructions for air-to-ground communication as well as handovers. We implement our secure avionic constructions and evaluate their performance on resource-constrained devices to account for resource limitations of on-board avionic components. We design our avionic handover construction

integrating our formalism for handovers, focusing on reducing the computational overhead of complex cryptographic operations. We formally prove the cryptographic security of our avionic protocol constructions within appropriate models capable of capturing their post-quantum security.

Contents

List of Figures

List of Tables

Nomenclature

BPSec Acronyms

BPA Bundle Protocol Agent

BPSec Bundle Protocol Security

DTN Delay Tolerant Network

BCB Bundle Confidentiality Block

BIB Bundle Integrity Block

SN Source Node

IN Intermediate Node

DN Destination Node

Handover Acronyms

HO Secure Handover Scheme

S Source

T Target

U User

PQAG **Acronyms**

A Aircraft

G Ground Station

ATC Air Traffic Control

ADS-B Automatic Dependant Surveillance-Broadcast

CPDLC Controller-Pilot Data Link Communications

FCAG-SIG Fully classical instantiation with ECC-DH and classical EdDSA signatures

PQAG Post Quantum Air-to-Ground

PQAG-KEM Hybrid instantiation with Kyber

PQAG-KEM-FQ Post-quantum instantiation with Kyber

PQAG-SIG-Ky-Di Hybrid instantiation with Kyber+ECC-DH and and post-quantum signature scheme Dilithium

PQAG-SIG-Ky Hybrid instantiation with Kyber and classical EdDSA signatures

PQAG-SIG-Mc Hybrid instantiation with McEliece and classical EdDSA signatures

Cryptographic Primitives & Acronyms

KEM Key Encapsulation Mechanism

KEX Key Exchange

KDF Key Derivation Function

MAC Message Authentication Code

KIND Key Indistinguishability

HAKE Hybrid Authenticated Key Exchange

PQC Post-quantum Cryptography

AKA Authenticated Key Exchange

Acknowledgments

This thesis is, improbably, the result of years of work, gallons of coffee, occasional bursts of inspiration, and a generous serving of chaos. First and foremost, I owe an immense debt of gratitude to Ben, my supervisor, who swooped in and rescued me two years into this PhD odyssey when I was unceremoniously abandoned by my original supervisors. To be fair, their abrupt departure might have been a blessing in disguise, but I would not say the whole ordeal gave my self-esteem the caffeine hit it desperately needed. Ben, your patience despite my refusal to even feign interest in understanding formal security proofs, my general snark, lack of respect for academic decorum, and occasional temper tantrums (that would make a toddler proud) has been nothing short of heroic. Truly, you were the saving grace of my PhD journey, even if you did slightly abandon me toward the end. (No hard feelings, though—I get it, your lofty career ambitions wait for no one). If I didn't fear that my abandoning the PhD would bruise your record as a first-time supervisor, I might not have finished. I hope you appreciate the weight of that selfless sacrifice. I will forever cherish our debates about cinema and the joy of poking holes in each other's arguments—though your bad internet connection and crappy webcam during remote meetings were cinematic experiences I could have done without. Despite your vehement refusal to change your ways—even when you're objectively wrong—I will always relish the camaraderie we shared, notably in our penchant for mid-meeting digressions into gossip, which, of course, was always a welcome respite.

A massive thanks to John, my line manager and perpetual supervisor savior. John, I owe you for stepping in every time a supervisor jumped ship—three times, to be precise. You've bailed me out so many times, you might want to consider changing your job title to "Academic Emergency Responder".

This PhD was also powered by an obscene amount of caffeine, a doting mother whose blind faith and unhealthy conviction in my abilities kept me going, and a father who couldn't say no to her and thus begrudgingly funded my ridiculous adventures far beyond anything

their humble lives prepared them for.

I also owe my sanity (or what's left of it) to Koutetsu Karate and Chris, for providing a socially acceptable way to channel my PhD-induced rage, to Emma and the Zumba gang for keeping the party animal in me alive in the middle of dreary Sheffield. Anna thank you for being my paler, saner and kinder twin and matching my freak, and Areeb, your perpetual existential crises were a weirdly comforting reminder that my life was somehow always marginally better. A big thank you to everyone, past and present, who crossed my path and, whether knowingly or unknowingly, helped me stumble my way to this moment—you are proof (no pun intended!) that every encounter—chance or otherwise—has a purpose. Special thanks to the queer community for being my home, my haven, and a constant reminder that I am part of something vibrant, beautiful, and *resilient*. You taught me that giving up is *not* in our nature, even when surrounded by the perpetual misery of rejected manuscripts and relentless deadlines.

Last but not least, I would like to extend my deepest (and most ironic) gratitude to the academic publishing process for teaching me invaluable lessons in patience, humility, and, let's be honest, sheer absurdity. Across two papers, I was rejected five times for reasons ranging from "not good enough" to "good enough, but wrong venue," to "too much theoretical foundation," to "not enough theoretical foundation." And let's not forget that one anonymous reviewer's adamant insistence on citing what was presumably their own unrelated work, which had absolutely no bearing on our research. Even the victories were Pyrrhic. A special shoutout to the reviewer who waited until we'd slogged through "major revisions" to unleash a laundry list of new demands during the "minor revisions" phase—because nothing says academic integrity like moving the goalposts. Your impeccable sense of timing, requiring these revisions during the Christmas holidays while I was on leave in another country, was truly inspiring. And to the journal that took a year to move from "accepted" to actually accepted: you've set a new standard for snail-like efficiency. Truly impressive. To these anonymous reviewers, thank you for reinforcing my belief that academia is where narcissism and pedantry collide in a seismic fashion. I did eventually get all my contribution papers accepted—because of my deep queer urge to slay in the face of extreme adversity—but the entire experience has left a bitter taste in my mouth. In the end, being a "published" author brought no joy, just the quiet relief that it's finally over.

In conclusion, this thesis was made possible by caffeine, stubbornness, a dash of chaos, and a community of people (and situations) that I both love and side-eye with suspicion. To all of you—thank you, sincerely.

To Dayani,
Without your unwavering, fanatical, and often unfounded blind faith in me (and in yourself), none of this would have been possible.

Part I

Introduction

Chapter 1

Motivation

For any nation-state the aerospace domain constitutes a crucial component of its critical national infrastructures which is defined by the US Department of Homeland Security as infrastructures "whose assets, systems, and networks, whether physical or virtual, are considered so vital that their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof" [44]. The aerospace domain having components of high reliability organizations (HROs)— such as air traffic control (ATC) systems for terrestrial avionic communication or satellites in the interplanetary internet— requires meticulous arrangements to accommodate for their hypercomplex operations and communications with compressed time frames and tight coupling of various systems. However, these systems have historically been largely isolated and reliant on analogue and legacy infrastructures. Consequently, any perceived attackers targeting such tightly monitored systems require high technical acumen, close proximity to targeted infrastructure and the ability to afford the large expenses associated with the acquisition of state-of-the-art equipment. Outwardly, these reasons collectively render the likelihood of the occurrence of such an attack next to negligible.

However, starting in the 1980s this status-quo of the aviation industry operating in relative isolation controlled by analog and legacy infrastructures of military origin started to gradually shift towards a commercial and digital era. The shift has been fairly slow owing to the lengthy and tedious standardization processes but at the time of writing multiple state organizations in Europe (EUROCONTROL) and the US (Federal Aviation Administration-FAA) are actively involved in the standardization and structuring of future communication infrastructures (FCI) for the aviation industry with digital communication

as its central component. The primary reasons fueling the necessity of this transformation can be associated with a plethora of factors; the drastic increase in air traffic load; global air traffic control (ATC) staff shortage; adverse environmental effects of the aviation industry; cost effectiveness and; advanced surveillance and safety capabilities.

Notwithstanding all the benefits of these much-needed transformations this process simultaneously subverts the previous security assumptions about fortitude of the avionic communication infrastructure. By replacing the previous analog and legacy military-grade infrastructures with digital media and commercial-off-the-shelf products, the aviation industry will no longer be able to consider itself isolated; nor rely on the assumption that a highly advanced attacker is unlikely. The integration of digital technologies as components of its critical operations, without careful considerations regarding their security implication, will drastically change the context of the current threat landscape of the aviation ecosystem. Further exacerbating the situation is the highly reliable nature of the aviation industry, meaning that an attack on one of its components will create a cascading effect that cripples the entire infrastructure with the possibility of a catastrophic event. A think paper published by EUROCONROL clearly demonstrates that cyberattacks against aviation systems are on the rise [4]. Therefore, within the current geopolitical climate, where state-funded attackers possess advanced capabilities and significant resources, the digitalization of the aviation industry presents a critical security challenge. Designed with safety as its priority rather than security, this industry faces heightened risks if its security implications are not carefully addressed. Thus, in order to mitigate these threats, it is imperative to implement appropriate defense mechanisms without further delay.

Concurrently, the renewed interest in space exploration by multiple stakeholders ranging from nation-states to business enterprises has created an urgent need for efficient and secure space communication. However, secure communication in non-terrestrial space presents its own set of unique challenges. Non-terrestrial internet is characterized by exceptionally long distances between nodes, irregular connectivity, high error rates, and asymmetric data links which distinguishes interplanetary internet from its terrestrial counterpart. This discrepancy coupled with the heightened interest in space research has urged NASA in collaboration with other space agencies to spearhead the development of protocols for the non-terrestrial interplanetary internet (IPN) [62]. The design objective that underlies the construction of these non-terrestrial protocols is to imitate their dependence on the terrestrial internet by mitigating the limitations of the deep space domain. Delay-tolerant network (DTN) protocols were borne out of this need [28, 52] which provide reliability guarantees for deep space

communication through a store-and-forward mechanism. This enables DTN protocols to circumnavigate the high-latency, volatility and the sporadic nature inherent to deep space networks. However, historically these network designs are kept closed source, leading to limited formal cryptographic analysis of the security offered by them. One of the few public protocols used in space networking is the Bundle Protocol, which is secured by Bundle Protocol Security (BPSec), an Internet Engineering Task Force (IETF) standard. There currently does not exist a model that appropriately captures the security of BPSec and as such its security remains unverified.

We set out to address the lack of formalization and stronger security guarantees within the context of secure aerospace communication. To this end, we start by formalizing handovers as a unique primitive in Part II. While not limited to aviation, handover protocols are commonly used during avionic data communications. Next, in Part III, we turn our attention to the security of deep space communication and undertake the first analysis of BPSec, building a flexible secure channel model that captures the security goals stated in the IETF standard. Finally, in Part IV of this thesis, we return to avionic communication and propose novel protocol constructions with stronger notions of security for next generation aviation. Additionally, in Part I Chapter 2 we provide an introduction to all the cryptographic primitives and assumptions that we have utilized for designing and capturing the theoretical security of protocols contained within this thesis. We conclude this thesis with Chapter 8, where we discuss key insights from our work and outline potential directions for future research. We will now summarize the contributions of this thesis.

Formalizing Secure Handovers (Chapter 3) As an aircraft moves from one geographic location to another during the course of its journey, the continuity of its communication session should be maintained until the end of its journey. In such environments, it is desirable for devices to securely extend their connection from one network to another, often referred to as a handover. In this work we introduce the first cryptographic formalization of secure handover schemes. We leverage our formalization to propose path privacy, a novel security property for handovers that has hitherto remained unexplored. We further develop a syntax for secure handovers, and identify security properties appropriate for secure handover schemes. Finally, we introduce a generic handover scheme that captures all the strong notions of security we have identified, combining our novel path privacy concept with other security properties characteristic to existing handover schemes, demonstrating the robustness and versatility of our framework. This generic scheme demonstrates that

simultaneously achieving strong notions of compromise resilience, key-indistinguishability, authenticity and path privacy are possible. It is not intended as a drop-in replacement for a specific aviation handover scheme, but a generic construction that achieves the strongest variants of security that can be downgraded as necessary for the setting. For instance, the property of path privacy is counter-intuitive to the use case of commercial aviation where the trajectory of an aircraft is regularly tracked for safety reasons. However, a military aircraft or a state-owned aircraft carrying high-profile individuals, may undoubtedly benefit from the additional location-privacy guarantees of path privacy.

Parts of this chapter have been accepted for publication in Alnashwan, A., Dowling, B., ***Wimalasiri, B.*** *(2025). Path Privacy and Handovers: Preventing Insider Traceability Attacks During Secure Handovers. 38th IEEE Computer Security Foundations Symposium (CSF).*

Security Analysis of Deep Space Protocols (Chapter 4) Space networking has become an increasing area of development with the advent of commercial satellite networks such as those hosted by Starlink and Kuiper, and increased satellite and space presence by governments around the world. We undertake a first analysis of BPSec, one of the few public protocols used in space networking. The BPSec secure channel environment, security expectations, and functionality are quite unlike traditional secure channel protocols, creating a non-trivial environment for analysis. We formalize the complex goals of BPSec and provide an analysis of the protocol. This includes a novel flexible secure channel model and proof corresponding to the type of security BPSec claims to offer. We identify issues in BPSec and outline normative security goals that it cannot assure. Furthermore, we provide recommendations for strengthening the security guarantees within the intended design constraints, introducing StrongBPSec, and provide a stronger model and analysis to match.

Parts of this chapter have been accepted for publication in Dowling, B., Hale, B., Tian, X., ***Wimalasiri, B.*** *(2024). Cryptography is Rocket Science: Analysis of BPSec. IACR Communications in Cryptology, 1(4).*

Secure Protocols for Next Generation Aviation (Chapters 5, 6 & 7) The rapid digitization of aviation communication and its dependent critical operations demand secure protocols that address domain-specific security requirements within the unique functional constraints of the aviation industry. These secure protocols must deliver robust protection against current and potential future attackers, taking into account the aviation community's

inherent complexity, resistance to frequent upgrades, and strict safety and cost constraints. In Chapter 5, we begin by presenting our framework KEX that formalizes key exchange constructions, which we leverage for the construction and formal analysis of our secure aviation handover PQAG-HO scheme described in Chapter 7. We then modify the HAKE hybrid key exchange framework, originally proposed by **?**], to integrate our KEX formalism, while simultaneously simplifying its construction. Next, in Chapter 6 we propose two quantum-secure hybrid key exchange protocols (PQAG-KEM and PQAG-SIG) to secure communication between aircrafts in-flight and ground stations. PQAG-KEM leverages post-quantum and classical Key Encapsulation Mechanisms (KEMs) to ensure the hybrid security of the protocol against classical as well as future quantum adversaries. PQAG-SIG, alternatively, uses quantum-safe digital signatures to achieve authentication security. We instantiate our proposed protocols with different post-quantum algorithms to understand their real-world applicability to the resource-constrained ecosystem of avionic communication. Furthermore, we formally analyze the security of our new PQAG protocols in a strong hybrid key exchange framework using our modified HAKE model discussed in Chapter 5. In Chapter 7 we turn our attention to handovers for avionic communication. We propose PQAG-HO, a robust quantum-secure handover mechanism tailored for resource-constrained avionic environments. The design of PQAG-HO minimizes computational overhead associated with complex cryptographic operations by integrating our KEX and HO formalisms. Furthermore, we instantiate and implement PQAG with appropriate cryptographic primitives to evaluate its performance and formally analyze its security.

Parts of this chapter have been accepted for publication in Dowling, B., ***Wimalasiri, B.*** *(2024). Quantum-Secure Hybrid Communication for Aviation Infrastructures. IEEE Transactions on Dependable and Secure Computing, doi: 10.1109/TDSC.2024.3483448.*

Chapter 2

Notation and Preliminaries

In this chapter, we introduce the basic notation used throughout this thesis and revisit some fundamental concepts and general cryptographic building blocks. Preliminaries specifically relevant to each contribution will be discussed in their respective chapters.

In subsequent chapters relating to the contributions of this thesis, we use game-based proof techniques to associate the difficulty of breaking the overall protocol to the difficulty of breaking the underlying cryptographic assumption. Cryptographic assumptions measure the computational difficulty required to breach the formal security objectives of a fundamental primitive. These assumptions are typically formalized as a game played between a well-defined challenger and a black-box adversary. The difficulty of breaking these assumptions is indicated by the adversary's advantage in winning conditions set by the challenger. We define a successful adversary as one that has a non-negligible advantage (or probability) of winning the game.

2.1 Notation

We begin by defining some common notation used throughout this thesis.

We use various typefaces to represent different types of objects: participants in security games (such as an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$), adversarial queries like $\mathsf{Corrupt}$ and $\mathsf{Compromise}$, protocol and per-session *variables*, security notions such as eufcma, and typewriter fonts for `constants`.

We denote by $\mathbb{N}$ the natural numbers as a set of non-negative integers, by $\mathbb{R}$ the real numbers, and by $\emptyset$ an empty set. A set $\mathbb{X} = \{1, ..., n\} = [1, n]$ indicates the finite set

of all integers between 1 and n inclusive. We use $\{0,1\}^\lambda$ to indicate the set of all λ-bit length integers, where λ is a security parameter within an asymptotic setting, and $\{0,1\}^*$ to indicate the set of all arbitrary-length bit strings. We implicitly define as $\mathcal{A}(1^\lambda)$ the unary representation 1^λ of λ to an algorithm $\mathcal{A}$. We denote with $x \xleftarrow{\$} \mathbb{X}$ the act of sampling an element x from $\mathbb{X}$ uniformly at random and we write $x \leftarrow y$ for the assignment of value y to the variable x. For a probabilistic algorithm $\mathcal{A}$, $b \leftarrow \mathcal{A}(x)$ denotes the output b of algorithm $\mathcal{A}$ when $\mathcal{A}$ takes as input random coins and x. We use $\perp$ as a generic failure symbol.

2.2 Cryptographic Building Blocks and Assumptions

In this section we briefly revisit some fundamental cryptographic concepts commonly utilized during the course of this thesis.

2.2.1 Security Experiments, Adversaries and Oracles

Security experiments or "games" are often used as a tool to formally define the security of cryptosystems. These games generally capture the interaction between two algorithms: a *challenger* and an *adversary*. These games follow a series of interactions between the two algorithms dictated by the specific cryptosystem tested and the latitude provided to the adversary to carry out their actions.

During the course of these security games an adversary often accesses an *oracle* in addition to the inputs and random bits. The concept of an oracle is that of an idealized black-box which returns a value on an input specified by the adversary, which is meant to model the fact that in practice an algorithm might have access to the answers to these queries without any constraints on the way they are computed or acquired.

At the end of a security game, the *challenger* algorithm returns a bit indicating result of the game— whether the *adversary* "won the game" by breaking the system or not. For a given game, the *success probability* of an adversary refers to the probability of the adversary winning the game against the challenger. The crypstosystem tested is generally deemed secure if there exists no adversary that succeeds with non-negligible probability in winning the game specified by the given security definition. The primary objective of these security games is to encompass a broad spectrum of attacks. Generally, the broader this spectrum, the more realistic the attack model becomes, making it increasingly challenging to design efficient cryptosystems.

We often discuss the advantage of an adversary $\mathcal{A}$ in winning a security game. In particular, we define a function $f : \mathbb{N} \to \mathbb{R}_{0,1}$ (where $\mathbb{R}_{0,1}$ is the set of all real numbers from 0 to 1 inclusive) to be negligible in x if for all positive integers $c > 0$ there exists a value $n_c > 0$ such that for all inputs $\lambda > n_c, f(x) < \frac{1}{n_c}$. We say that such a function is negligible for a given input security parameter λ.

2.2.2 Running Time

For each possible input size λ, a probabilistic polynomial-time algorithm (or PPT) is a polynomial-time uniform family of classical Boolean circuits. A probabilistic algorithm is said to be PPT if it strictly runs in time polynomial in the length of its inputs. Similarly, a quantum polynomial-time algorithm (or QPT) can be defined as a polynomial-time uniform family of quantum circuits, each composed of gates that may perform general admissible operations, chosen from some finite, universal set.

2.2.3 Privacy, Anonymity & Unlinkability

In Chapter 3 of this thesis, we formalize the notions of *unlinkability* and *path privacy* for our *secure handover* framework. Our definition of privacy is informed by existing notions of privacy in authenticated key exchange schemes [8, 58], though the distinctive features of our construction prevent a direct one-to-one alignment. More specifically, we formulate our notion of *path privacy* based on the observation that *handovers* represent a distinct cryptographic primitive, separate from traditional key exchange constructions. As a result, retrofitting existing privacy notions— originally designed for key exchange— fails to capture the nuanced requirements of secure handovers.

Goldberg et al. [58] argues that *anonymity* is an enabling technology for *privacy*; when captured within constructions such as two–party key exchange schemes, it can ensure complete obscurity of a party's identity, even from its communication peer. The notion of *identity hiding*, which ensures that a party's identity is concealed from external entities but *not* from its communication peer, is frequently studied as a weaker security property in comparison to *anonymity*. Widely deployed secure communication protocols, such as TLS 1.3 [102], adopt the weaker notion of identity hiding rather than full anonymity in order to support mutual authentication (or unilateral authentication in TLS 1.3), since achieving authentication under complete anonymity introduces significant complexity.

However, as Arfaoui et al. [8] argues, capturing only the notion of *identity hiding* overlooks the risk of partial information leakage— for example, the ability to link an *anonymous* user to two distinct sessions. To address this, modern privacy-preserving protocols typically strengthen their security guarantees by incorporating *unlinkablity* in lieu of basic *identity hiding*. The notion of *unlinkability* describes the property in which an attacker cannot distinguish whether two items of interest are related or not [99]. As Goldberg et al. [58] describes, in the context of key exchange, *unlinkability* prevents an attacker from distinguishing whether two independent key exchange sessions are linked to the same party (potentially anonymous) or to two distinct parties.

Drawing from such existing literature but diverging on specifics, we model our notions of *unlinkability* and *path privacy* to capture the subtleties of secure handovers. Our notion of *unlinkability* concerns only an *external attacker* who attempts to link an *anonymous* user across distinct sessions. Conversely, our notion of *path privacy* considers an *insider attacker*— specifically, the user's current and next communication partners— who attempt to compromise the location privacy of a user that has revealed their identity to these peers (and is thus unanonymized). We further elaborate on and formalize these notions in Chapter 3.

2.2.4 Symmetric Cryptography

In the following, we present details of symmetric cryptographic primitives that underlie the work contained within this thesis. Broadly, cryptography is categorized into two fundamental groups: symmetric and asymmetric. Symmetric cryptographic primitives rely on a single shared key for performing their operations, while asymmetric primitives utilize a pair of keys: a public key that can be openly shared and a secretly maintained private key.

Hash Functions

A hash function is a commonly used cryptographic building block that compresses an arbitrary length input message into a fixed length digest. Hash functions play an integral role in the design of primitives such as digital signature schemes, message authentication codes and key derivation functions as well as being commonly used in protocol constructions. Following their wider use and within our constructions, we exclusively consider hash functions to be unkeyed, and provide the property of collision-resistance. We briefly describe the property of collision resistance as follows [107].

Given a hash function $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^\lambda$ where λ is the fixed output length of Hash it should be difficult to find two distinct points pt_1, pt_2 in the domain of Hash such that $\mathsf{Hash}(pt_1) = \mathsf{Hash}(pt_2)$ but $pt_1 \neq pt_2$.

Now we proceed to define a collision-resistant hash function.

Definition 1 (Collision-resistant Hash function)**.** *A hash function is a deterministic algorithm* $\mathsf{Hash} : \{0,1\}^* \to \{0,1\}^\lambda$ *which, given a bit string* m*, outputs a hash value* $w = \mathsf{Hash}(m)$ *in the space* $\{0,1\}^\lambda$*. We define the advantage of a probabilistic polynomial-time (PPT) algorithm* $\mathcal{A}$ *in breaking the collision-resistance security of the hash function* Hash *as* $\mathsf{Adv}^{\mathsf{coll},\lambda}_{\mathsf{Hash}}(\mathcal{A}) = \Pr[\mathsf{Hash}(m) = \mathsf{Hash}(m') : (m, m') \leftarrow \mathcal{A}, m \neq m']$*. We say that a hash function* Hash *is collision-resistant if for all efficient PPT adversary* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{coll},\lambda}_{\mathsf{Hash}}(\mathcal{A})$ *is negligible in the security parameter* λ*.*

Our PQAG-HO construction discussed in Chapter 7 leverages the security of Hash functions to verify message transcripts. Furthermore, all constructions proposed within this thesis utilize other Hash-based constructions such as MAC and HKDF, which we discuss next.

Message Authentication Codes

Message Authentication Codes (MACs) allow two parties with a shared secret key to authenticate messages by generating a MAC tag τ that verify the integrity of their corresponding messages. In security protocols, MACs are commonly used to authenticate parties and ensure the integrity of the exchanged message transcripts. The security of a MAC scheme is typically described with regards to its unforgeability: existential, selective or universal. Existential unforgeability implies that it is challenging for an adversary to produce any forged message-tag pair (m, τ) such that $\mathsf{Vfy}(k, m, \tau)$ outputs 1. We outline the syntax for MAC schemes and their standard security notions, including existential and strong unforgeability under chosen-message attacks [69].

Definition 2 (Message Authentication Code (MAC) Scheme)**.** *A message authentication code (*MAC*) scheme is a tuple of algorithms* $\mathsf{MAC} = \{\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vfy}\}$ *where:*

- $\mathsf{KGen}(1^\lambda) \xrightarrow{\$} k$ *is a probabilistic key generation algorithm taking as input a security parameter* 1^λ *and returning a symmetric key* k*.*
- $\mathsf{Tag}(k, m) \xrightarrow{\$} \tau$ *is a (possibly) probabilistic algorithm that takes as input a symmetric key* k *and an arbitrary message* m *from the message space* $\mathcal{M}$ *and returns a* tag τ*.*

– $\mathsf{Vfy}(k, m, \tau) \rightarrow \{0, 1\}$ *is a deterministic verification algorithm that takes as input a symmetric key* k*, a message* m*, and a* tag τ*.* Vfy *outputs a bit* b*, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid*.*

We say a MAC scheme provides correctness if for every λ*,* $k \leftarrow \mathsf{KGen}(1^\lambda)$ *and* m*, it holds that* $\mathsf{Vfy}(k, m, \mathsf{Tag}(k, m)) = 1$*.*

Definition 3 (Existential and strong unforgeability of MACs)**.** *Let* $\mathsf{MAC} = (\mathsf{KGen}, \mathsf{Tag}, \mathsf{Vfy})$ *be a MAC scheme. The experiment for existential and strong unforgeability* $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda)$ *against adversary* $\mathcal{A}$ *is defined as follows:*

1. *The challenger samples* $k \xleftarrow{\$} \mathcal{K}$
2. *The adversary may adaptively query the challenger; for each query value* m_i *the challenger responds with* $\tau_i = \mathsf{Tag}(k, m_i)$
3. *The adversary outputs a pair of values* (m^*, τ^*) *such that* $(m^*, \tau^*) \notin \{(m_0, \sigma_0), \ldots (m_i, \sigma_i)\}$

The adversary $\mathcal{A}$ *wins the game if* $\mathsf{Tag}(k, m^*) = \tau^*$*, producing a* tag *forgery. We define the advantage of* $\mathcal{A}$ *in breaking the existential and strong unforgeability property of a MAC* MAC *under chosen-message attack to be:*

$$\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = \Pr\left(\mathsf{Tag}\left(k, m^*\right) = \tau^*\right)$$

We say that a MAC scheme is post-quantum sufcma*-secure if, for all QPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*, and is classically* sufcma*-secure if, for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

We rely on the existential and strong unforgeability property of MAC schemes for the security of our constructions presented in Chapters 3, 6 and 7. Moreover, in Chapter 4, we leverage the sufcma security of MAC schemes for our analysis of the BPSec protocol instatiated with HMAC-SHA2.

HMAC-Key Derivation Function (HKDF)

The role of a key derivation function is to derive one or more cryptographically strong secret keys using some source of initial keying material, typically containing some good amount of randomness, but not distributed uniformly or for which an attacker has some

partial knowledge[73]. The work of Krawczyk [73] proposes an extract-then-expand KDF that consists of a randomness extractor and a variable-length output pseudorandom function (PRF). The randomness extractor is presumed to produce "close to random outputs" on inputs sampled from the source key material distribution. The output of the randomness extractor is used as the input for the PRF along with a length parameter and context information string (label) which may be null. As the real-world instantiation of this extract-then-expand KDF, Krawczyk [73] proposes the use of HMAC to serve as the random extractor as well as the PRF. In our proposed PQAG protocols discussed in Chapters 6 and 7, the keys obtained from key encapsulation mechanisms are used as the source key material for the HKDF.Extract function. The resulting key is in turn used as the input for the HKDF.Expand function. We define prfs and their security as follows.

Definition 4 (prf Security)**.** *A pseudo-random function family is a collection of deterministic functions* $\mathsf{PRF} = \{\mathsf{PRF}_\lambda : \mathcal{K} \times \mathcal{I} \to \mathcal{O} : \lambda \in \mathbb{N}\}$*, one function for each value of* λ*. Here,* $\mathcal{K}$*,* $\mathcal{I}$*,* $\mathcal{O}$ *all depend on* λ*, but we suppress this for ease of notation. Given a key* k *in the keyspace* $\mathcal{K}$ *and a bit string* $m \in \mathcal{M}$*,* PRF_λ *outputs a value* y *in the output space* $\mathcal{O} = \{0,1\}^\lambda$*. We define the security of a pseudo-random function family in the following game between a challenger* $\mathcal{C}$ *and a PPT adversary* $\mathcal{A}$*, with* λ *as an implicit input to both algorithms:*

1. $\mathcal{C}$ *samples a key* $k \xleftarrow{\$} \mathcal{K}$ *and a bit* b *uniformly at random.*
2. $\mathcal{A}$ *can now query* $\mathcal{C}$ *with polynomially-many distinct* m_i *values, and receives either the output* $y_i \leftarrow \mathsf{PRF}_\lambda(k, m_i)$ *(when* $b = 0$*) or* $y_i \xleftarrow{\$} \{0,1\}^\lambda$ *(when* $b = 1$*).*
3. $\mathcal{A}$ *terminates and outputs a bit* b'*.*

We say that $\mathcal{A}$ *wins the* PRF *security game if* $b' = b$ *and define the advantage of a algorithm* $\mathcal{A}$ *in breaking the* pseudo-random function security *of a PRF family* PRF *as* $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda) = |2 \cdot \Pr(b' = b) - 1|$*. We say that* PRF *is post-quantum* prf*-secure if for all QPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*. We say that* PRF *is secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

HMAC is a dual PRF, which informally refers to a PRF that remains secure if either the key material or the label carries entropy. HMAC has been shown to be a secure MAC under the assumption it is a dual PRF [26], and Bellare and Lysyanskaya [17] have given a generic validation of the dual PRF assumption for HMAC and therefore HKDF.

We now turn to describe dual-prf security for pseudorandom functions [17]. On a high-level, a PRF achieves dual-prf security if the PRF retains its prf security when keyed with either the k input or the m.

Definition 5 (dual-prf Security)**.** *Let* PRF *be a PRF family. We define a second PRF family* $\mathsf{PRF}^{\mathsf{dual}} = \{\mathsf{PRF}^{\mathsf{dual}}_{\lambda} : \mathcal{I} \times \mathcal{K} \to \mathcal{O} : \lambda \in \mathbb{N}\}$ *by setting* $\mathsf{PRF}^{\mathsf{dual}}_{\lambda}(m,k) = \mathsf{PRF}_{\lambda}(k,m)$*. We define the advantage of* $\mathcal{A}$ *in breaking the* dual-prf *security of* PRF *as* $\mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda) = \max\{\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda), \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF}^{\mathsf{dual}},\mathcal{A}}(\lambda)\}$*. We say that* PRF *is post-quantum* dual-prf*-secure PRF family if, for all QPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*, and* PRF *is* dual-prf*-secure PRF family if, for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

A puncturable pseudo-random function is a special instance of a pseudo-random function (PRF) (Definition 4), that facilitates the computation of punctured keys, which prohibits evaluation on inputs that have been punctured. When used within symmetric authentication schemes(e.g. MACs), PPRF prevents replay attacks and guarantee forwards secrecy by rendering expired keys unusable.

We refer to the definition of puncturable pseudo-random functions and its security from [110], but restrict our attention to PPRFs with deterministic puncturing algorithms as defined by [13].

Definition 6 (pprf Security)**.** *A puncturable pseudorandom function* $\mathsf{PPRF} = (\mathsf{SetUp}, \mathsf{Eval}, \mathsf{Punc})$ *is a triple of algorithms with three associated sets; the secret-key space* $\mathcal{K}$*, the domain* $\mathcal{X}$ *and the range* $\mathcal{Y}$*. We describe the algorithm as follows:*

- $\mathsf{Setup}(1^{\lambda}) \xrightarrow{\$} sk : \mathsf{Setup}$ *is a probabilistic algorithm that takes as input a security parameter* λ *and outputs an evaluation key* $sk \in \mathcal{K}$*.*
- $\mathsf{Eval}(sk, x) \to y/\perp : \mathsf{Eval}$ *is an evaluation algorithm that accepts as input the secret key* sk *and an element* $x \in \mathcal{X}$ *and outputs* $y \in \mathcal{Y}$ *or, to indicate failure,* $\perp$*.*
- $\mathsf{Punc}(sk, x) \to sk' : \mathsf{Punc}$ *is a deterministic puncturing algorithm that accepts as input the secret key* sk *and an element* $x \in \mathcal{X}$*, and outputs an updated secret key* $sk' \in \mathcal{K}$*.*

PPRF *is correct if for every subset* $x_1, \ldots, x_n = \mathcal{S} \subseteq \mathcal{X}$ *and all* $x \in \mathcal{X} \backslash \mathcal{S}$*, we have that* $\Pr\left[\mathsf{Eval}(sk_0, x) = \mathsf{Eval}(sk_n, x) : \begin{smallmatrix} sk_0 \leftarrow \mathsf{Setup}(1^{\lambda}); \\ sk_i = \mathsf{Punc}(sk_{i-1}, x_i) \text{ for } i \in [n]; \end{smallmatrix}\right] = 1$.

In order to guarantee the security of our StrongHO construction we require our PPRF function to be *invariant to puncturing*. That is to say, the puncturing is "commutative" and, the order in which one punctures the key does not affect the resulting secret key. Aviram et al. [12] formally defines *invariant puncturing* as follows:

Definition 7 (Invariant PPRF)**.** *A* PPRF *is invariant to puncturing if for all keys* $k \in \mathcal{K}$ *and all elements* $x_0, x_1 \in \mathcal{X}$*,* $x_0 \neq x_1$ *it holds that*

$$\mathsf{Punc}(\mathsf{Punc}(k, x_0)x_1) = \mathsf{Punc}(\mathsf{Punc}(k, x_1)x_0)$$

Our security experiments for PPRF closely follow that of [12], which we have presented in Figure 2.1 .

$\underline{\mathsf{Exp}^{\mathsf{random}}_{\mathcal{A},\mathsf{PPRF}}(\lambda)}$

1: $k \xleftarrow{\$} \mathsf{Setup}(1^\lambda), b \xleftarrow{\$} \{0,1\}, \mathcal{Q} := \emptyset$
2: $x^* \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Eval}}(k,.)}(1^\lambda)$ where $\mathcal{O}_{\mathsf{Eval}}(k, x)$ behaves like Eval, but sets $\mathcal{Q} := \mathcal{Q} \cup \{x\}$.
3: $y_0 \xleftarrow{\$} \mathcal{Y}, y_1 := \mathsf{Eval}(k, x^*), k := \mathsf{Punc}(k, x^*)$
4: $b^* \xleftarrow{\$} \mathcal{A}(k, y_b)$
5: return 1 if $b = b^* \wedge x^* \notin \mathcal{Q}$
6: return 0

Figure 2.1: Adaptive-random PPRF *security experiment.*

Authenticated Encryption & Authenticated Encryption with Associated Data

Authenticated Encryption (AE) and Authenticated Encryption with Associated Data (AEAD) are symmetric primitives that aim to provide both confidentiality as well as integrity of messages exchanged. Bellare and Namprempre [18] first proposed Authenticated Encryption which formalized two notions of integrity: *integrity of plaintexts* (INT-PTXT) formalized that an adversary is unable to forge a valid ciphertext that decrypts correctly for a previously unseen message; and *integrity of ciphertexts* (INT-CTXT) formalized that an adversary is unable to forge a valid ciphertext, even if the underlying plaintext has already been revealed. Rogaway [106] extended this work and introduced Authenticated Encryption with *Associated Data*. This work recognized that in real-world applications some data such as routing information benefit from integrity-protection but should not be encrypted alongside a payload it is attached to. AEAD provides a mechanism that enables authenticating such data as a whole while providing privacy only for the plaintext attached under

indistinguishaility of plaintexts ind-cpa security. Briefly, ind-cpa security guarantees that an attacker cannot distinguish between the ciphertexts of two chosen plaintexts, even after observing multiple encryptions under the same key.

In this thesis, we have leveraged the security of AE within the design of our StrongHO construction discussed in Chapter 3. We integrate the privacy and integrity guarantees provided by AEAD within our PQAG-HO constructions described in Chapter 7. We further utilize the security of AEAD for the analysis of the BPSec protocol presented in Chapter 4.

Definition 8 (Authenticated Encryption)**.** *An AE scheme* AE *is a triple of algorithms* $\mathsf{AE} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with an associated keyspace* $\mathcal{K}$*, nonce* $\mathcal{N} \in \{0,1\}^n$*, and message space* $\mathcal{M} \in \{0,1\}^*$*. These sets all depend on the security parameter* λ*. We denote by* $\mathsf{AE.KGen}(\lambda) \rightarrow k$ *a key generation algorithm that takes as input* λ *and outputs a key* $k \in \mathcal{K}$*. We denote by* $\mathsf{AE.Enc}(k, N, M) \rightarrow C$ *the AE encryption algorithm that takes as input a key* $k \in \mathcal{K}$*, a nonce* N*, and a message* $M \in \mathcal{M}$ *and outputs a ciphertext* $C \in \{0,1\}^*$*. We denote by* $\mathsf{AE.Dec}(k, N, C)$ *the AE decryption algorithm that takes as input a key* $k \in \mathcal{K}$*, a nonce* N*, and a ciphertext* C *and returns a string* M'*, which is either in the message space* $\mathcal{M}$ *or a distinguished failure symbol* $\perp$*. Correctness of an* AE *scheme requires that* $\mathsf{AE.Dec}(k, N, \mathsf{AE.Enc}(k, N, M)) = M$ *for all* k, M *in the appropriate space.*

Definition 9 (AE Security)**.** *Let* AE *be an AE scheme, and* $\mathcal{A}$ *a PPT algorithm with input* λ *and access to an oracle* $\mathsf{Enc}(.,.,.)$*. This oracle, given input* (N, M)*, outputs* $\mathsf{Enc}(k, N, M)$ *for a randomly selected key* $k \in \mathcal{K}$*. We say that* $\mathcal{A}$ *forges a ciphertext, if* $\mathcal{A}$ *outputs* (N, C) *such that* $\mathsf{Dec}(k, N, C) \rightarrow M \neq \perp$ *and* (N, M) *was not queried to the oracle. We define the advantage of a PPT algorithm* $\mathcal{A}$ *in forging a ciphertext as* $\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE},\mathcal{A}}(\lambda)$*. We say that an AE scheme* AE *is* ae*-secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ae}}_{\mathsf{AE},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

We further provide a fine-grained AE-AUTH security definition which we leverage to guarantee *path privacy* within our StrongHO construction.

Definition 10 (AE-AUTH Security)**.** *Let* AE *be an AE scheme, and* $\mathcal{A}$ *a PPT algorithm with input* λ *and access to an oracle* $\mathsf{Enc}(.)$*. This oracle, given input* (N, M)*, outputs* $\mathsf{Enc}(k, N, M)$ *for a randomly selected key* $k \in \mathcal{K}$*. We say that* $\mathcal{A}$ *forges a ciphertext if* $\mathcal{A}$ *outputs* C *such that* $\mathsf{Dec}(k, N, C) \rightarrow M \neq \perp$ *and* M *was not queried to the oracle. We define the advantage of a PPT algorithm* $\mathcal{A}$ *in forging a ciphertext as* $\mathsf{Adv}^{\mathsf{AUTH}}_{\mathsf{AE},\mathcal{A}}(\lambda)$*. We say that an AE scheme* AE *is* AE-AUTH *secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{AUTH}}_{\mathsf{AE},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Next, we describe an extended and widely adopted variant of authenticated encryption: authenticated encryption with associated data (AEAD).

Definition 11 (Authenticated Encryption with Associated Data)**.** *An AEAD scheme AEAD is a tuple of algorithms* $\mathsf{AEAD} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with an associated keyspace* $\mathcal{K}$*, nonce* $\mathcal{N} \in \{0,1\}^n$*, message space* $\mathcal{M} \in \{0,1\}^*$ *and headers* $\mathcal{H} \in \{0,1\}^*$*. These sets all depend on the security parameter* λ*. We denote by* $\mathsf{AEAD.KGen}(\lambda) \to k$ *a key generation algorithm that takes as input* λ *and outputs a key* $k \in \mathcal{K}$*. We denote by* $\mathsf{AEAD.Enc}(k, N, H, M)$ *the AEAD encryption algorithm that takes as input a key* $k \in \mathcal{K}$*, a nonce* $N \in \mathcal{N}$*, a header* $H \in \mathcal{H}$ *and a message* $M \in \mathcal{M}$ *and outputs a ciphertext* $C \in \{0,1\}^*$*. We denote by* $\mathsf{AEAD.Dec}(k, N, H, C)$ *the AEAD decryption algorithm that takes as input a key* $k \in \mathcal{K}$*, a nonce* $N \in \mathcal{N}$*, a header* $H \in \mathcal{H}$ *and a ciphertext* C *and returns a string* M'*, which is either in the message space* $\mathcal{M}$ *or a distinguished failure symbol* $\perp$*. Correctness of an* AEAD *scheme requires that* $\mathsf{AEAD.Dec}(k, N, H, (\mathsf{AEAD.Enc}(k, N, H, M)) = M$ *for all* k, N, H, M *in the appropriate space.*

Definition 12 (AEAD Security)**.** *Let* AEAD *be an AEAD scheme, and* $\mathcal{A}$ *a PPT algorithm with input* λ *and access to an oracle* $\mathsf{Enc}(.,.,.)$*. This oracle, given input* (N, H, M)*, outputs* $\mathsf{Enc}(k, N, H, M)$ *for a randomly selected key* $k \in \mathcal{K}$*. We say that* $\mathcal{A}$ *forges a ciphertext, if* $\mathcal{A}$ *outputs* (N, H, C) *such that* $\mathsf{Dec}(k, N, H, C) \to M \neq \perp$ *and* (N, H, M) *was not queried to the oracle. We define the advantage of a PPT algorithm* $\mathcal{A}$ *in forging a ciphertext as* $\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}}(\lambda)$*. We say that an AEAD scheme* AEAD *is* aead *secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Moreover, for the security analysis of the BPSec protocol discussed in Chapter 4, we require a more fine-grained definition for AEAD security which differentiates between AEAD-PRIV and AEAD-AUTH. Accordingly, we provide a suitable definition below, which closely follows the work of Rogaway and Shrimpton [105].

Definition 13 (AEAD-PRIV SECURITY)**.** *Let* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *be* AEAD*-scheme with length function* ℓ*. Let* $\$(\cdot,\cdot,\cdot)$ *be an oracle that, upon input returns* (N, H, M)*, returns a random string of* $\ell(|M|)$ *bits. The advantage of adversary* $\mathcal{A}$ *in breaking* AEAD PRIV*-security is defined as*

$$\mathsf{Adv}^{\mathsf{PRIV}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{Enc}_K(.,.,.)} \implies 1\right] - \Pr\left[\mathcal{A}^{\$(.,.,.)} \implies 1\right].$$

We say an AEAD scheme is AEAD-PRIV *secure if there is exists no efficient PPT adversary $\mathcal{A}$ that can distinguish between a real ciphertexts and a uniformly random string of the same length.*

Definition 14 (AEAD-AUTH SECURITY)**.** *Let $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ be* AEAD*-scheme, and let $\mathcal{A}$ be an adversary with access to an orcale $\mathsf{Enc}_K(\cdot,\cdot,\cdot)$. We say that $\mathcal{A}$* forges *(for this key K and on some particular run) if $\mathcal{A}$ outputs $(N, H, \mathcal{C})$ where $\mathsf{Dec}_K^{N,H}(\mathcal{C}) \neq \perp$ and $\mathcal{A}$ did not query $\mathsf{Enc}_K^{N,H}(M)$ to the encryption oracle. We say that an AEAD scheme* AEAD *is* auth*-secure if for all PPT algorithms $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{AEAD},\mathcal{A}}(\lambda)$ is negligible in the security parameter λ.*

Next we describe the formalism for deterministic authenticated encryption (DAE) and define its security which closely follow the definitions of Rogaway and Shrimpton [108] who introduced this primitive. We leverage the security of DAE for the analysis of the BPSec protocol in Chapter 4. Specifically, BPSec instantiates with *key-wrapped* keys and we analyze the security of this instantiation using DAE to verify the indistinguishability of the keys used for *key wrapping*.

Definition 15 (Deterministic Authenticated Encryption)**.** *A deterministic authenticated encryption scheme* DAE *is a tuple $\mathsf{DAE} = \{\mathcal{K}, \mathsf{Enc}, \mathsf{Dec}\}$ with associated key space $\mathcal{K}$, message space $\mathcal{M} \subseteq \{0,1\}^*$ and headers $\mathcal{H} \subseteq \{0,1\}^{**}$. The key space $\mathcal{K}$ is a set of strings or infinite strings endowed with a distribution. For a practical scheme there must be a probabilistic algorithm that samples from $\mathcal{K}$, and we identify this algorithm with the distribution it induces. We denote by $\mathsf{DAE.Enc}_K(H, M)$ the deterministic* DAE *encryption algorithm that takes as input a key $K \in \mathcal{K}$, a header $H \in \mathcal{H}$ and a message $M \in \mathcal{M}$ and outputs either a ciphertext $C \in \{0,1\}^*$ or a distinguished failure symbol $\perp$. We denote by $\mathsf{DAE.Dec}_K(H, C)$ the deterministic DAE decryption algorithm that takes as input a key $K \in \mathcal{K}$, a header $H \in \mathcal{H}$ and a ciphertext C and returns a string M', which is either in the message space $\mathcal{M}$ or a distinguished failure symbol $\perp$. We assume that $M \in \mathcal{M} \implies \{0,1\}^{|M|} \subseteq \mathcal{M}$. The ciphertext space is $C = \{\mathsf{DAE.Enc}_K(H, M) : K \in \mathcal{K}, H \in \mathcal{H}, M \in \mathcal{M}\}$. Correctness of an* DAE *scheme requires that $\mathsf{DAE.Dec}_K(H, (\mathsf{DAE.Enc}_K(H, M)) = M$ for all K, H, M in the appropriate space.*

We note that in our construction presented in Figures 4.11, 4.12 and 4.13 we have adopted the notation KW.Enc and KW.Dec instead of DAE.Enc and DAE.Dec, respectively, as defined above. We wanted our notation to adhere to the vocabulary of the BPSec Default Security Context [29] and thus WLOG we substituted DAE with KW within our

construction. Next, we describe the security of DAE schemes. On a high level, DAE-security ensures ind-cpa confidentiality of the underlying plaintext and authenticity of the ciphertexts. We leverage the security of DAE to prove that there exists no such adversary that can forge a DAE ciphertext without being detected. We provide this in Definition 16.

Definition 16 (DAE Security)**.** *Let* $\mathsf{DAE} = \{\mathcal{K}, \mathsf{Enc}, \mathsf{Dec}\}$ *be a DAE scheme with header space* H*, message space* M*, and expansion function* e*. The advantage of adversary* $\mathcal{A}$ *in breaking* dae*-security is defined as*

$$\mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{A}}(\lambda) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{Enc}_K(.,.),\mathsf{Dec}_K(.,.)} \implies 1\right] - \Pr\left[\mathcal{A}^{\$(.,.),\perp(.,.)} \implies 1\right].$$

Upon query $H \in \mathcal{H},\ M \in \mathcal{M}$*,* $\mathcal{A}$*'s random oracle* $\$(.,.)$ *returns a random string of length* $|M| + e(H, M)$*. The* $\perp(.,.)$ *oracle returns* $\perp$ *on every input. We assume that* $\mathcal{A}$ *does not ask* (H, C) *of its right oracle if some previous left oracle query* (H, M) *returned* C*; does not ask* (H, M) *of its left oracle if some previous right-oracle query* (H, C) *returned* M*; does not ask left queries outside of* $H \times M$ *; and does not repeat a query.*

2.2.5 Asymmetric Cryptographic

Asymmetric cryptography relies on the use of distinct public and private key pair, operating under the assumption that the private key cannot be feasibly derived from the corresponding public key.

Digital Signature Schemes

Digital signature schemes enable a party with a publicly available public key to sign messages using the corresponding private key. These signed messages may then be verified by others with the corresponding public-key, and confirm that the signed message was sent by the private key's owner. We integrate the authentication guarantees provided by digital signatures, that bind signed messages to unique owners, within our StrongHO and PQAG-SIG constructions presented in Chapters 3 and 6 respectively.

Definition 17 (Digital Signature Schemes)**.** *A digital signature* (SIG) *scheme is a tuple of algorithms* $\mathsf{SIG} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vfy}\}$ *where:*

- $\mathsf{KGen} \xrightarrow{\$} (pk, sk)$ *is a probabilistic key generation algorithm taking input a security parameter* λ *and returning a public key* pk *and a secret key* sk*.*

- $\mathsf{Sign}(sk, m) \xrightarrow{\$} \sigma$ *is a probabilistic algorithm that takes as input a secret key* sk *and an arbitrary message* m *from the message space* $\mathcal{M}$ *and returns a signature* σ*.*
- $\mathsf{Vfy}(pk, m, \sigma) \rightarrow \{0, 1\}$ *is a deterministic algorithm that takes as input a public key* pk*, an message* m *and a signature* σ *and returns bit* $b \in \{0, 1\}$*.*

We require correctness of a digital signature scheme SIG*. Specifically, for all* $(pk, sk) \xleftarrow{\$} \mathsf{SIG.KGen}$*, we have* $\mathsf{SIG.Vfy}(pk, m, \mathsf{SIG.Sign}(sk, m)) = 1$*.*

We describe the security of digital signatures against both classical and post-quantum adversaries, capturing the existential and strong unforgeability under chosen message attack.

Definition 18 (Classical sufcma Security of Digital Signature Schemes)**.** *Let* $\mathsf{SIG} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vfy}\}$ *be a classical digital signature scheme. Security is formulated via the following game that is played between a challenger* $\mathcal{C}$ *and PPT algorithm* $\mathcal{A}$*:*

1. *The challenger samples* $pk, sk \xleftarrow{\$} \mathcal{K}$
2. *The adversary may adaptively query the challenger; for each query value* m_i *the challenger responds with* $\sigma_i = \mathsf{Sign}(sk, m_i)$
3. *The adversary outputs a pair of values* (m^*, σ^*) *such that* $(m^*, \sigma^*) \notin \{(m_0, \sigma_0), \ldots (m_i, \sigma_i)\}$

The adversary $\mathcal{A}$ *wins the game if* $\mathsf{Vfy}(pk, m^*, \sigma^*) = 1$*, producing a signature forgery. We define the advantage of* $\mathcal{A}$ *in breaking the existential unforgeability property of a digital signature scheme* SIG *under chosen-message attack to be:*

$$\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) = \Pr\left(\mathsf{Vfy}\left(pk, m^*, \sigma^*\right) = 1\right)$$

We say that classical SIG *scheme is* sufcma*-secure if, for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Definition 19 (Post-quantum sufcma Security of Digital Signature Schemes)**.** *Let* $\mathsf{SIG} = \{\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vfy}\}$ *be a post-quantum digital signature scheme. Security is formulated via the following game that is played between a challenger* $\mathcal{C}$ *and QPT algorithm* $\mathcal{A}$*:*

1. *The challenger samples* $pk, sk \xleftarrow{\$} \mathcal{K}$
2. *The adversary may adaptively query the challenger; for each query value* m_i *the challenger responds with* $\sigma_i = \mathsf{Sign}(sk, m_i)$

3. *The adversary outputs a pair of values* (m^*, σ^*) *such that* $(m^*, \sigma^*) \notin \{(m_0, \sigma_0), \ldots (m_i, \sigma_i)\}$

The adversary $\mathcal{A}$ *wins the game if* $\mathsf{Vfy}(pk, m^*, \sigma^*) = 1$*, producing a signature forgery. We define the advantage of* $\mathcal{A}$ *in breaking the existential unforgeability property of a digital signature scheme* SIG *under chosen-message attack to be:*

$$\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{A}}(\lambda) = \Pr\left(\mathsf{Vfy}\left(pk, m^*, \sigma^*\right) = 1\right)$$

We say that post-quantum SIG *scheme is* sufcma*-secure if, for all QPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Key Encapsulation Mechanism

Key encapsulation mechanisms (or KEMs) is a cryptographic primitive that enables one party in possession of another party's public key to securely transport a secret key to the second party. Coretti et al. [45] describes KEMs as a key-exchange protocol that only transmits a single message which integrates symmetric encryption to achieve public-key encryption of messages of arbitrary length. Given the ubiquity of KEMs within the post-quantum paradigm as well as their adaptability to model classical key exchange primitives, key encapsulation mechanisms have been integrated as a building block across multiple constructions discussed in this thesis. They act as a key component of our $\mathsf{StrongHO}$ construction presented in Chapter 3 as well as within our quantum-secure hybrid protocol schemes discussed in Chapters 6 and 7.

Definition 20 (Key Encapsulation Mechanism)**.** *A key encapsulation mechanism (KEM) is a triple of algorithms* $\mathsf{KEM} = \{\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps}\}$ *with an associated keyspace* $\mathcal{K}$*. We describe the algorithms below:*

- $\mathsf{KGen}\lambda \xrightarrow{\$} (pk, sk) : \mathsf{KGen}$ *is a probabilistic algorithm that takes as input the security parameter* λ *and returns a public/secret key pair* (pk, sk)*.*
- $\mathsf{Encaps}(pk) \xrightarrow{\$} (c, k) : \mathsf{Encaps}$ *is a probabilistic algorithm that takes as input a public key* pk *and outputs a ciphertext* c *as well as a key* $k \in \mathcal{K}$*.*
- $\mathsf{Decaps}(sk, c) \rightarrow (k) : \mathsf{Decaps}$ *is a deterministic algorithm that takes as input a secret key* sk *and a ciphertext* c *and outputs a key* $k \in \mathcal{K}$*, or a failure symbol* $\perp$*.*

KEM *is correct if for all* (pk, sk) *such that* $\mathsf{KGen}(\lambda) \xrightarrow{\$} (pk, sk)$*, and* (c, k) *such that* $\mathsf{Encaps}(pk) \xrightarrow{\$} (c, k)$*, it holds that* $\mathsf{Decaps}(sk, c) = k$*.*

We define KEMs and their ind-cpa security against both classical and post-quantum algorithms as follows.

Definition 21 (Classical ind-cpa Security of KEMs)**.** *Let* $\mathsf{KEM} = \{\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps}\}$ *be a classical KEM. We define the* ind-cpa *security of a key encapsulation mechanism in the following game played between a challenger* $\mathcal{C}$ *and PPT adversary* $\mathcal{A}$*.*

1. $\mathcal{C}$ *generates a public-key pair* $\mathsf{KGen}(\lambda) \xrightarrow{\$} (pk, sk)$
2. $\mathcal{C}$ *generates a ciphertext and key* $\mathsf{Encaps}(pk) \xrightarrow{\$} (c, k_0)$
3. $\mathcal{C}$ *samples a key* $k_1 \xleftarrow{\$} \mathcal{K}$ *and a bit* b *uniformly at random.*
4. $\mathcal{A}$ *is given* (pk, c, k_b) *and outputs a guess bit* b'

We say that $\mathcal{A}$ *wins the* ind-cpa *security game if* $b' = b$ *and define the advantage of an algorithm* $\mathcal{A}$ *in breaking the* ind-cpa *security of a key encapsulation mechanism* KEM *as* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = |2 \cdot \Pr(b' = b) - 1|$*. We say that a classical* KEM *is* ind-cpa*-secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Definition 22 (Post-quantum ind-cpa Security of KEMs)**.** *Let* $\mathsf{KEM} = \{\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps}\}$ *be a post-quantum KEM. We define the* ind-cpa *security of a key encapsulation mechanism in the following game played between a challenger* $\mathcal{C}$ *and QPT adversary* $\mathcal{A}$*.*

1. $\mathcal{C}$ *generates a public-key pair* $\mathsf{KGen}(\lambda) \xrightarrow{\$} (pk, sk)$
2. $\mathcal{C}$ *generates a ciphertext and key* $\mathsf{Encaps}(pk) \xrightarrow{\$} (c, k_0)$
3. $\mathcal{C}$ *samples a key* $k_1 \xleftarrow{\$} \mathcal{K}$ *and a bit* b *uniformly at random.*
4. $\mathcal{A}$ *is given* (pk, c, k_b) *and outputs a guess bit* b'

We say that $\mathcal{A}$ *wins the* ind-cpa *security game if* $b' = b$ *and define the advantage of an algorithm* $\mathcal{A}$ *in breaking the* ind-cpa *security of a key encapsulation mechanism* KEM *as* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = |2 \cdot \Pr(b' = b) - 1|$*. We say that a post-quantum* KEM ind-cpa*-secure if for all QPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Now we strengthen our assumptions by defining ind-cca security for KEMs. Compared to an ind-cpa attacker, an ind-cca adversary can encrypt messages of their choosing as well as access decryptions/decapsulations of ciphertexts of their choosing even after the challenge has been received. A special variant of ind-cca is ind-1cca in which the adversary is allowed to make only one decryption/decapsulation query.

Definition 23 (ind-cca Security of KEMs)**.** *A key encapsulation mechanism (KEM) is a triple of algorithms* $\mathsf{KEM} = \{\mathsf{KGen}, \mathsf{Encaps}, \mathsf{Decaps}\}$ *with an associated keyspace* $\mathcal{K}$*, as described above.*

We define the ind-cca *security of a key encapsulation mechanism in the following game played between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

1. $\mathcal{C}$ *generates a public-key pair* $\mathsf{KGen}(\lambda) \xrightarrow{\$} (pk, sk)$
2. $\mathcal{C}$ *generates a ciphertext and key* $\mathsf{Encaps}(pk) \xrightarrow{\$} (c, k_0)$
3. $\mathcal{C}$ *samples a key* $k_1 \xleftarrow{\$} \mathcal{K}$ *and a bit* b *uniformly at random.*
4. $\mathcal{A}$ *is given* (pk, c, k_b)
5. *The adversary may adaptively query the challenger; for each query value* $ctxt_i$ *the challenger responds with* $k_i = \mathsf{Decaps}(sk, ctxt_i)$
6. *The adversary outputs a guess bit* b'

We say that $\mathcal{A}$ *wins the* ind-cca *security game if* $b' = b$ *and define the advantage of an algorithm* $\mathcal{A}$ *in breaking the* ind-cca *security of a key encapsulation mechanism* KEM *as* $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda) = |2 \cdot \Pr(b' = b) - 1|$*. We say that a post-quantum* KEM *is* ind-cca*-secure if for all QPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*. We say that a classical* KEM *is* ind-cca*-secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Public Key Encryption

Public key encryption facilitates secure communication without the need to agree on a shared secret beforehand. Public key encryption is an asymmetric primitive that leverages two distinct keys: *sender* encrypts the message with *receiver*'s *public key* and the *receiver* decrypts the ciphertext with the corresponding *private key.* Public keys are typically tied to the identity of the corresponding owner and can be shared over an insecure channel. Private keys are maintained secretly and is only known to their owner.

Definition 24 (Public Key Encryption)**.** *A public key encryption (*PKE*) scheme is a tuple of algorithms* $\mathsf{PKE} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *where:*

- $\mathsf{KGen}(1^\lambda) \xrightarrow{\$} (pk, sk)$ *is a probabilistic key generation algorithm taking input a security parameter* λ *and returning a public key* pk *and a secret key* sk*.*
- $\mathsf{Enc}(pk, m) \xrightarrow{\$} c$ *is a probabilistic algorithm that takes as input a public key* pk *and an arbitrary message* m *from the message space* $\mathcal{M}$ *and returns a ciphertext* c*.*

– $\mathsf{Dec}(sk, c) \rightarrow m$ *is a deterministic algorithm that takes as input a secret key* sk *and a ciphertext* c *and returns a message* m*.*

We require correctness of a PKE *scheme. Specifically, for all* $(pk, sk) \xleftarrow{\$} \mathsf{PKE.KGen}$*, we have* $\mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m)) = m$*.*

Our StrongHO protocol presented in Chapter 3 integrates public key encryption as a building block in its design. We rely on the *key indistinguishability* notion of a PKE scheme (ikcca) for the security of our StrongHO construction, particularly with regards to guaranteeing the notion of *path privacy*. Informally, the ikcca-security of a PKE scheme guarantees that an adversary $\mathcal{A}$ cannot distinguish whether a ciphertext c was encrypted using a real public key pk or a uniformly random public key $\widetilde{pk}$ even after adaptively querying a PKE decryption oracle.

Definition 25 (Key Indistinguishability of PKE)**.** *Let* $\mathsf{PKE} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *be a PKE scheme. The experiment* $\mathsf{Adv}^{\mathsf{ikcca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *for indistinguishability of keys under chosen ciphertext attack security is formulated via the following game that is played between a challenger* $\mathcal{C}$ *and an algorithm* $\mathcal{A}$*:*

1. *The challenger samples* $(pk_0, sk_0) \xleftarrow{\$} \mathsf{KGen}(1^\lambda), (pk_1, sk_1) \xleftarrow{\$} \mathcal{K}(1^\lambda)$ *and submits* (pk_0, pk_1) *to the adversary.*
2. *The adversary may adaptively query the challenger; for each query value* (c_i, d) *the challenger responds with* $m_i = \mathsf{Dec}(sk_d, c_i)$*.*
3. *The adversary outputs a value* x*; the challenger samples a bit* $b \xleftarrow{\$} \{0, 1\}$ *and returns* $c^* \xleftarrow{\$} \mathsf{Enc}(pk_b, x)$*.*
4. *The adversary may adaptively query the challenger; for each query value* (c_i, d) *if* $c_i = c^*$ *the challenger responds with* $\perp$*, else the challenger responds with* $m_i = \mathsf{Dec}(sk_d, c_i)$*.*
5. *The adversary eventually terminates and outputs* b'*.*

The adversary $\mathcal{A}$ *wins the game if* $b' = b$*. We define the advantage of* $\mathcal{A}$ *in breaking the key indistinguishability property of a public key encryption scheme* PKE *under chosen-ciphertext attack to be:*

$$\mathsf{Adv}^{\mathsf{ikcca}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = |\Pr\left(b' = b\right) - \frac{1}{2}|$$

We say that PKE *is* ikcca*-secure if, for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{ikcca}}_{\mathsf{PKE},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

Part II

Formalization of Handovers

Chapter 3

Secure Handover Protocols

We begin this chapter by establishing the uniqueness of handovers through investigating existing literature on secure handover schemes and the security properties they claim to deliver. We then introduce a universal formalized framework that captures HO schemes as a unique primitive. We further introduce the notion of path privacy and construct security experiments that capture distinct security properties for HOs, thus setting it apart from other similar primitives such as key exchanges. We also construct a generic strong HO scheme that achieves our defined path privacy property alongside other security goals. We further provide a formal analysis of security for our strong construction within the HO framework. Additionally, we investigate the universal applicability of our formalization by illustrating its ability to model various existing HO constructions.

3.1 Introduction

The topic of securely handling communication sessions during their transition between zones has garnered much attention recently, notably owing to the prevalence of 5G communication within current mobile networks. Currently, the 3GPP/5G handover protocol [97] remains the sole widespread implementation of a secure handover protocol. Briefly, a 5G handover involves the transfer of an existing connection from one node (or base station within the 5G architecture) to another as a result of user devices moving between different zones or their current zone becoming unavailable due to other simultaneous connections [97]. To elaborate, the *transfer of an existing communication* here refers to the process of seamlessly moving an active communication session from one network node (or base station) to another. This

process ensures uninterrupted service as the device transitions between different coverage areas, known as zones or cells, within the 5G network architecture.

However, there exists another example of an (insecure) handover scheme that is commonly adopted across the globe: Controller-Pilot Data Link Communications (CPDLC) is a protocol that facilitates communication between the Air Traffic Control (ATC) stations and aircrafts over a digital datalink medium. As an aircraft travels from one geographic location to another, CPDLC facilitates for an automatic transference of its current communication session from one ATC station (or ground station) to another, eliminating the requirement to re-establish a new session every time an aircraft enters a subsequent geographic zone [65].

Despite numerous proposals for innovative handover schemes [76], there exists no formal framework defining handovers as a distinct primitive with unique functional requirements and security properties. Most literature on handover schemes is modeled after the 5G protocol [97, 94], with little discussion on systematically defining the functional goals of a handover or distinguishing handover schemes from other primitives like key exchange protocols.

3.1.1 Security of Handovers

Both 5G handover and CPDLC face vulnerabilities to various proposed attacks. Gupta et al. [60] highlight a de-synchronisation attack on the 5G handover protocol using a rogue base station to disrupt communication and enable denial-of-service attacks. Basin et al. [15] analyze the 5G-AKA protocol, which the 5G handover extends, finding that it allows attackers to impersonate base stations and exploit vulnerabilities to make another user responsible for service usage charges. They also note that the 5G-AKA protocol only protects user privacy from passive attackers. In their comprehensive analysis of the 5G handover, Peltonen et al. [97] identify risks in transmitting session and intermediate keying parameters over a secure interface, which, if compromised, jeopardizes 5G handover security. They further observe that any compromise to these keys at any stage will compromise all future key derivations.

Unsurprisingly, CPDLC does not provide any security guarantees, since all communications are carried out over unencrypted and unauthenticated channels. Smailes et al. [115] explore practical attacks against CPDLC, focusing on impersonating ATC stations to aircrafts. They successfully hijack a session between an aircraft and a legitimate ATC station

during the handover phase. In their attack they trigger a false handover to a new ATC station by injecting messages into an existing CPDLC session. Their attacks may have catastrophic consequences, as CPDLC messages contain critical instructions as declaring emergencies, changing altitudes or changing speeds of an aircraft, which Smailes et al. [115] show can easily be hijacked and tampered with by an unsophisticated attacker. They highlight the practicality of these attack by launching them as an attacker stationed hundreds of kilometers away from their targets.

3.1.2 Secure Handovers, a Unique Primitive

While 5G handover achieves some security properties, we note that the attacks discussed here illustrate that these current handover deployments lack a cohesive security framework, and that some attacks exploit distinct functionalities of handovers. For instance, consider the CPDLC attack that triggers a false handover, or the ability to hijack sessions during the handover phases. Modeling handover schemes using existing frameworks such as key exchange constructions is insufficient to address these types of attacks, as key exchanges are not inherently designed to secure the transition of existing sessions between participating entities. Formalizing distinct notions of security for handover schemes clarifies security guarantees necessary during a handover phase of a protocol, preventing the possibility of attacks as demonstrated in [115]. We emphasize that previous work that model handovers as a variation of the *key exchange* primitive [6, 53] do not appropriately capture the intricacies of handovers.

We generically define a handover scheme HO as a cryptographic protocol executed between three parties: a user $\mathtt{U}$, a source $\mathtt{S}$, and a target $\mathtt{T}$. $\mathtt{U}$ is mobile, traveling between different zones and communicating with the station in that zone, much like a mobile phone travels between different base stations when their owner walks down the street. Handover schemes concern $\mathtt{U}$'s transition between these zones: specifically, we assume that $\mathtt{U}$ has previously communicated with the current station in their zone (which we denote the source $\mathtt{S}$), and wishes to continue their current communication session in the proceeding zone with the new station, target $\mathtt{T}$. A handover scheme allows $\mathtt{S}$ to communicate and authenticate sufficient information to $\mathtt{T}$, allowing $\mathtt{U}$ to continue their communication session with $\mathtt{T}$ without re-executing a full handshake between $\mathtt{U}$ and $\mathtt{T}$ - usually by establishing some shared secret key between the two.

3.1.3 Key Exchange vs Handover

Initially, it seems as though U and T could simply execute a key exchange protocol to authenticate each other and establish a shared secret key. Indeed, often these two primitives are discussed in an interchangeable manner, but this obscures the distinction between the two. A three-party key exchange protocol would require all parties to establish shared secret keys to achieve key indistinguishability for a given session. This is clearly different from HO, where S does not need to know the fresh secret established between U and T, and indeed S should *not* be able to compute this.

Additionally, a three-party key exchange protocol typically has symmetrical authentication relationships: each party is likely to authenticate to each other in a similar fashion. Conversely, within HO, there exists an asymmetry to the pre-established trust relationships. For example, neighboring stations are likely to know each other's public keys, but T is unlikely to know U's public keys. Additionally, S and U are likely to share pre-established symmetric keys due to some previous handover. Thus, the S node acts as a proxy of trust to independently authenticate U to T to transition its current communication session with U to T, in a manner that ensures the continuity of the original session.

The pre-establishment of keys, the asymmetric trust relationships and the continuity-preserving transition of communication collectively sets HO apart from key exchange protocols as a unique primitive. The widely-adopted 5G handover protocol [97] exemplifies all these characteristics that we have identified as unique to HO. Thus, we endeavor to address this gap and systemically treat HO, formalizing it as a distinct primitive and capture its security.

3.1.4 Path Privacy

Multiple instances of subscriber privacy violations and data breaches by cellular network operators have drawn significant attention in recent years. For example, the FCC recently fined AT&T, Sprint, T-Mobile, and Verizon millions of dollars for unlawfully sharing users' geolocation data with third parties, including prisons and bounty hunters, within their commercial programs [57, 46]. In one case, a bounty hunter hired by Motherboard [46] successfully tracked their target to a specific neighborhood in Queens, New York, just blocks away from their actual location.

Contrarily, in the case of commercial aircrafts their travel plans are often public information and is freely available often days in advance. However, with regards to stakeholders

in the military, government organizations or private businesses, there is a greater need for location privacy compared to their commercial counterparts. For instance, it may be required to keep the movement and communications of a certain business stakeholder flight private due to its potential impact on stock prices (e.g. mergers, acquisitions etc.) [85]. Stakeholders wishing to keep their flight movements and communication private may request to "block" their data from appearing on publicly available flight data websites (e.g. flightaware, flightradar24). However, due to the use of open and unencrypted channels for avionic communication it may yield these attempts at privacy futile [116] revealing sensitive information such as location data. In fact, the work of Strohmeier et al. [119] successfully maps the trajectory of a military aircraft that had blocked their data from appearing on public websites, but was still using using unencrypted communication channels. As such, we identify that some settings will require the network to conceal the migration trajectory of U as they move from one location to another. Particularly, since HOs are often used in a geographical context, HOs that do not conceal a user's footprint between nodes can leak information about a user's physical location.

Works that analyze the 5G handover protocol [53, 6] often capture anonymity against an external attacker, but do not consider the insider threat posed by various administrative nodes. Indeed, only anonymizing the identity of U from serving nodes (S or T) is insufficient to obfuscate their trajectory, since U's journey can still be linked to their anonymized identifier. Failing to achieve *path privacy* may also result in an elevated risk for stalking attacks. The use of affordable and commercially-off-the-shelf IoT devices has been gaining popularity as an accessible mechanism for stalking, as substantiated by the recent surge of stalking attacks using Apple airtags [92]. HOs that leak path information may further enable these attacks, and so we identify *path privacy* as a desired security notion to address this threat. To the best of our knowledge, there does not currently exist work that formally capture this security notion within the context of HO schemes, which we accomplish in our framework. We introduce *path privacy*, which leverages our formalized secure handovers primitive to provide a simpler and practical framework to introduce privacy guarantees into existing infrastructure with formal proofs verifying its security guarantees.

3.1.5 Formal Security Notions for Handovers

From our study of existing literature, we have identified several shared security goals that are deemed as desirable within secure HO schemes. Predominantly, the need for *key in-*

distinguishability for derived session keys between U and T and a necessity to maintain *user anonymity* as one moves between different networks have been identified as fundamental security requirements across multiple independent bodies of work [6, 53]. We implicitly capture *mutual authentication* as a core security property, to eliminate the threat of session-hijacking attacks against HO [115]. In Section 3.3 we propose a generic strong HO scheme StrongHO that achieves all our formalized security goals, demonstrating how to realize strong notions of security in a HO setting. This must not be interpreted as a drop-in replacement for CPDLC or 5G, but a generic construction that achieves the strongest variants of security that can be downgraded as necessary for the setting. In our construction in Figure 3.7, we focus on the optimal strongest level of security achievable within a HO construction, capturing notions of *forward-secrecy*, *mutual authentication*, *user unlinkability* and *path privacy*.

3.1.6 Related Work

The literature on secure handover schemes (HO) is saturated with self-identified handover schemes covering a wide range of contexts. While a significant percentage of studied HO schemes focus on 5G mobile communication [127, 95, 64, 53, 133, 40, 61, 6], handover schemes have also been proposed for cloud computing architectures [131], aviation [87, 72], urban air mobility (UAM) networks [76] and VANETs [117].

Broadly speaking, all works share structural similarities in their HO protocol constructions, and conform to a similar fundamental architecture. All works describe the setting where a user U leverages an authentication mechanism to establish some token or secret with another party (denoted the source S) node prior to the HO execution. The actual process of a handover requires S node acting as an intermediary to transition the responsibility of secure communication to the U node from itself to a tertiary party whom we call the target T node. This transition of communication from the S node to the T node is expected to be done in a seamless and efficient manner that preserves the continuity of communication while ensuring low computational overhead and latency.

Security Properties for Handovers

With respect to security properties, *mutual authentication* has been considered as an essential security property in all of the studied work. Within the three-party architecture of a HO construction— between a user U, a source S and a target T— *mutual authentication*

is often achieved in an asymmetric manner. For instance, often a source S independently authenticates with U and T and act as an intermediary that negotiates the secure transition of communication. Moreover, efficient revocation of users, *anonymity*, *unlinkability* and *perfect forward secrecy* have further been listed as desirable security properties in many of the studied works. The desirability of such properties as *anonymity* and *unlinkability* can be attributed to maintaining user privacy while *forward secrecy* ensures the confidentiality of past communications even after a current session key has been compromised. We list all the works studied in Table 3.1, describing claimed security properties and the status of their formal security analysis.

Provable Security for Secure Handovers

While a diverse selection of methodologies have been utilized to analyze the security of secure handover protocols, some work that introduces novel handover schemes do not formally analyze the protocols at all [131]. Indeed, while most works surveyed claim anonymity and unlinkability, only the works of [6] and [53] provide formal proofs to verify these security notions. However, their work is exclusively modeled after the 5G-HO protocol [97] and they do not capture the notion of *path privacy*. Furthermore, they model their handover protocols as key exchange schemes, thus failing to capture the uniqueness of HO as a primitive.

Peltonen et al. [97] formally analyze the security of the 5G-HO scheme in the symbolic model, using the verification tool Tamarin. Miller et al. [91] extend [97] to model *honest-but-curious* base stations who may collude to compromise the *forward secrecy* guarantees of 5G-HO. Blazy et al. [30] introduce a general framework for quantifying and comparing post-compromise security (PCS), and apply it to model the healing speed of sequential compositions of 5G-HO after a compromise. They observe that, depending on the specific 5G-HO sequence, PCS may take multiple hops to recover— or may not recover at all— without protocol modifications. Alnashwan et al. [6] and Fan et al. [53] propose security improvements for the standard 5G-HO protocol and analyze the security of their proposed schemes in the computational model. Norrman [94] models 5G-HO as a secure anycast channel [96] and comes closest to our work presented in this chapter, but is strongly tied to 5G-HO specifically, with assumptions of pre-existing secure-channels for communication and and a central *orchestrator* entities for service provision. The model of Norrman [94] cannot be easily adapted to model such constructions as HO schemes for aviation [65, 87] that do not rely on centralized entities for handover facilitation. Moreover, their work does

Work \ Properties	PFS	Key Indistinguishability	Unlinkability	Mutual Authentication	Anonymity	Source Privacy	Target Privacy
Son et al. [117]	●[AVISPA]	●[CP]	○	●[BAN/AVISPA]	○	○	○
Wang et al. [127]	●[Scyther]	○	○	●[BAN/Scyther]	○	○	○
Nyangaresi et al. [95]	○	○	○	○[BAN]	○	○	○
Huang and Qian [64]	○	○	○	○[BAN]	○	○	○
Fan et al. [53]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	○	○
Alnashwan et al. [6]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	○	○
Zhang et al. [133]	●[AVISPA]	●[AVISPA]	○	●[BAN/AVISPA]	○	○	○
Cao et al. [40]	●[Scyther]	●[Scyther]	○	●[BAN/Scyther]	○	○	○
Yang et al. [131]	○	○	○	○	○	○	○
Gupta et al. [61]	●[CP/AVISPA]	●[CP/AVISPA]	○	●[CP/AVISPA]	○	○	○
Kwon et al. [76]	●[AVISPA]	●[ROR]	○	●[BAN/AVISPA]	○	○	○
Maurer et al. [87]	●[Tamarin]	○	○	●[Tamarin]	○	○	○
Khan et al. [72]	○	○	○	○	○	○	○
Our StrongHO	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]

Table 3.1: Comparison of some proposed handover schemes and their security properties. ● Formal security proofs, ○ No formal security proofs, CP - Computational Proofs.

not capture *user unlinkability* and *path privacy* properties and instead focuses solely on confidentiality and integrity of data transmissions.

A significant number of examined works applies BAN-logic [37] to prove the properties of mutual authentication and key agreement in their proposed schemes. However, the suitability of BAN-logic as a framework to analyze the security of protocols has been contested and the works of Nessett [93], Lowe [79], and Boyd and Mao [32] capture serious security flaws in protocols proven secure under the BAN-logic. As such, in our study we consider work that solely analyze the security of their proposed schemes with BAN-logic as insufficient, and have categorized them as work providing no formal security proofs in Table 3.1. Conversely, the works of Cao et al. [40], Son et al. [117], Wang et al. [127], Zhang et al. [133] and Kwon et al. [76] combine BAN-logic with formal proofs obtained through automated security protocol verification tools, thus providing stronger security guarantees.

In this section we have critiqued the various security proofs presented in existing literature to formally verify the purported security of their respective HO schemes. We highlight that our StrongHO protocol described in Section 3.7 is the first to achieve all properties simultaneously. Next, we direct our attention to formalizing handovers as a unique primitive

and defining and capturing desirable secure properties for such handover shcemes.

3.2 Formalizing Secure Handovers

In this section we formalise our notion of secure handover protocols, explaining the expected functionality, phases and outputs. We follow by detailing the security goals that secure handover schemes can achieve. We give a brief explanation of each goal, and then describe the security model that captures each goal.

3.2.1 Handover Syntax

We consider a protocol that is executed between three parties: a user U, a source S and a target T. U has, in some previous interaction, established some shared secret state with S, and now wishes to leverage S's connection with T to establish some new authenticated shared state (potentially secret) with T. The reasoning behind the decision to limit our formalization to three parties is as follows; for a HO to occur U must at least transition from one S to one T; ours is a flexible approach capable of integrating any additional parties in existing HO-specific protocols, e.g. core network in 5G-HO can easily be abstracted into our S or T roles depending on whether they have an existing session with U or not; and our approach simplifies and generalizes parties participating in HO protocols. In general, a handover protocol HO has four distinct phases:

- A *setup* phase, where the protocol participants generate long-term secrets (e.g. digital signature key pairs); U and S generate some shared secret state and agree on some additional data *ad* that needs to be advocated to T (abstractly capturing an initial key exchange, or a previous handover);
- A *preparation* phase, where U and S interact to generate some material that allows S to authenticate information that will be used by the user to communicate to T. This preparation phase allows for the handover protocol to achieve *source* or *target privacy*;
- A *support* phase, where S and T interact and transfer the previous material; if a protocol construction aims to achieve *path privacy* this phase will be precluded;
- A *contact* phase, where U and T directly interact and execute a handover protocol together, authenticate each other and establish some shared secret state.

Thus, a handover protocol HO consists of a tuple of algorithms HO = {Gen, SGen,Setup, Prep, Supp, Cont}:

- $\mathsf{Gen}(1^\lambda) \xrightarrow{\$} (pk, sk, \mathsf{pid})$: Gen is a probabilistic algorithm independently run by all parties that takes a security parameter λ and outputs the long-term public key pair (pk, sk) and (potentially) identifiers of user (id), source $(spid)$, and target $(tpid)$. This algorithm broadly captures long-term key generation.
- $\mathsf{SGen}(1^\lambda) \xrightarrow{\$} (bk, \mathsf{id})$: SGen is a probabilistic algorithm run by U and S, which takes as input a secret parameter λ and outputs a (bootstrap) secret key bk and (potentially) identifiers of the user (id), source $(spid)$, and target $(tpid)$. SGen abstractly captures a user U that executes an authentication mechanism to establish some token or secret with another party (denoted the source S) prior to the HO execution.
- $\mathsf{Setup}(id_i, id_j, bk, sk_i, pk_j, \rho) \xrightarrow{\$} (st)$: Setup is a probabilistic algorithm run by all parties, which accepts as input the identifiers of the communicating parties of the current session (id_i, id_j), a shared secret key bk, (potentially) some long-term secret key of the executing party sk_i (where $i \in \{\mathtt{U}, \mathtt{S}, \mathtt{T}\}$), (potentially) long-term public key of the communicating party pk_j (where $j \in \{\mathtt{U}, \mathtt{S}, \mathtt{T}\}$) and the role ρ of the executing party and outputs an initial state st at the start of a HO transaction. Setup captures session management per protocol execution by explicitly generating the states maintained by parties.
- $\mathsf{Prep}(st, pk_\mathtt{T}, m) \xrightarrow{\$} (st', m)$: is a probabilistic interactive algorithm run by U and S, which takes as input the secret state st, potentially the long-term public key of T $pk_\mathtt{T}$, and (potentially) some input message m, and outputs updated state st' and (potentially) some output message m. This algorithm enables the realization of path privacy by allowing U to act as an intermediary that facilitates the mutual authentication of S and T.
- $\mathsf{Supp}(st', pk_j, pk_u, m) \xrightarrow{\$} (st, m')$: is a probabilistic interactive algorithm run by S and T, which takes as input the state st, (potentially) the long-term public key of the communicating party pk_j (where $j \in \{\mathtt{S}, \mathtt{T}\}$), the long-term public key of U $pk_\mathtt{U}$, and (potentially) some input message m, and outputs the updated state st' and (potentially) some output message m. Once Supp is completed the S node deletes all state data (st', m) pertaining to that session.
- $\mathsf{Cont}(st, pk_j, pk_\mathtt{S}, m) \xrightarrow{\$} (st', m')$: is a probabilistic interactive algorithm run by U and T, which takes as input the secret state st, the long-term public key of the

communicating party pk_j (where $j \in \{\mathtt{U}, \mathtt{T}\}$), the long-term public key of $\mathtt{S}$ $pk_{\mathtt{S}}$, and (potentially) some input message m, and outputs some updated state st' and (potentially) some output message m.

We give an execution of this process in Figure 3.1. We note that all secure HO constructions perform some Setup and at minimum need to execute either Prep+Supp or Prep+Cont.

The modular nature of our proposed framework provides a high level of flexibility that can be easily adapted to the specific requirements of any handover design. Apart from the initial Setup phase which abstracts away the prerequisites required for a HO execution, all other phases in our formalization can be added or subtracted according to the demands of the specific design. For instance, in our strong StrongHO scheme proposed in Section 3.3, we forgo the Supp phase in order to capture the property of *path privacy* in our construction. The proposed secure LDACS-HO [87] does not include a Contact phase since $\mathtt{S}$ acts as an intermediary throughout, forwarding messages between $\mathtt{U}$ and $\mathtt{T}$ and no direct communication takes place between $\mathtt{U}$ and $\mathtt{T}$ until the HO is completed. Our framework therefore provides a highly customizable and flexible structure that formalizes aspects of network limitations as seen with LDACS [87], while also encouraging the integration of stronger security notions as in our StrongHO construction discussion in Section [3.3]. In Figure 3.2, we illustrate the flexibility of our framework by mapping the existing 5G [97] and CPDLC HO [115] protocols and the proposed LDACS-HO [87] to our construction. Our mapping in Figure 3.2 highlights that, regardless of their specific design, any secure handover scheme can be modeled within our framework.

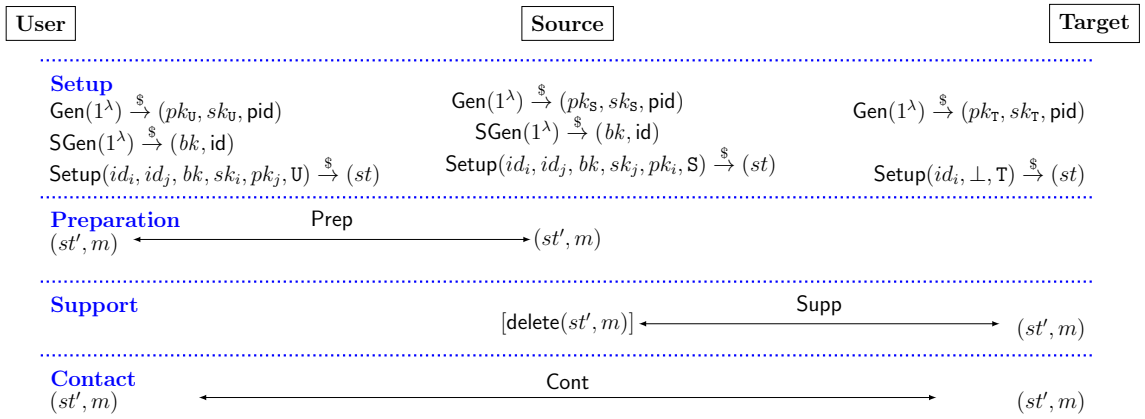


Figure 3.1: An expected execution of a secure HO protocol.

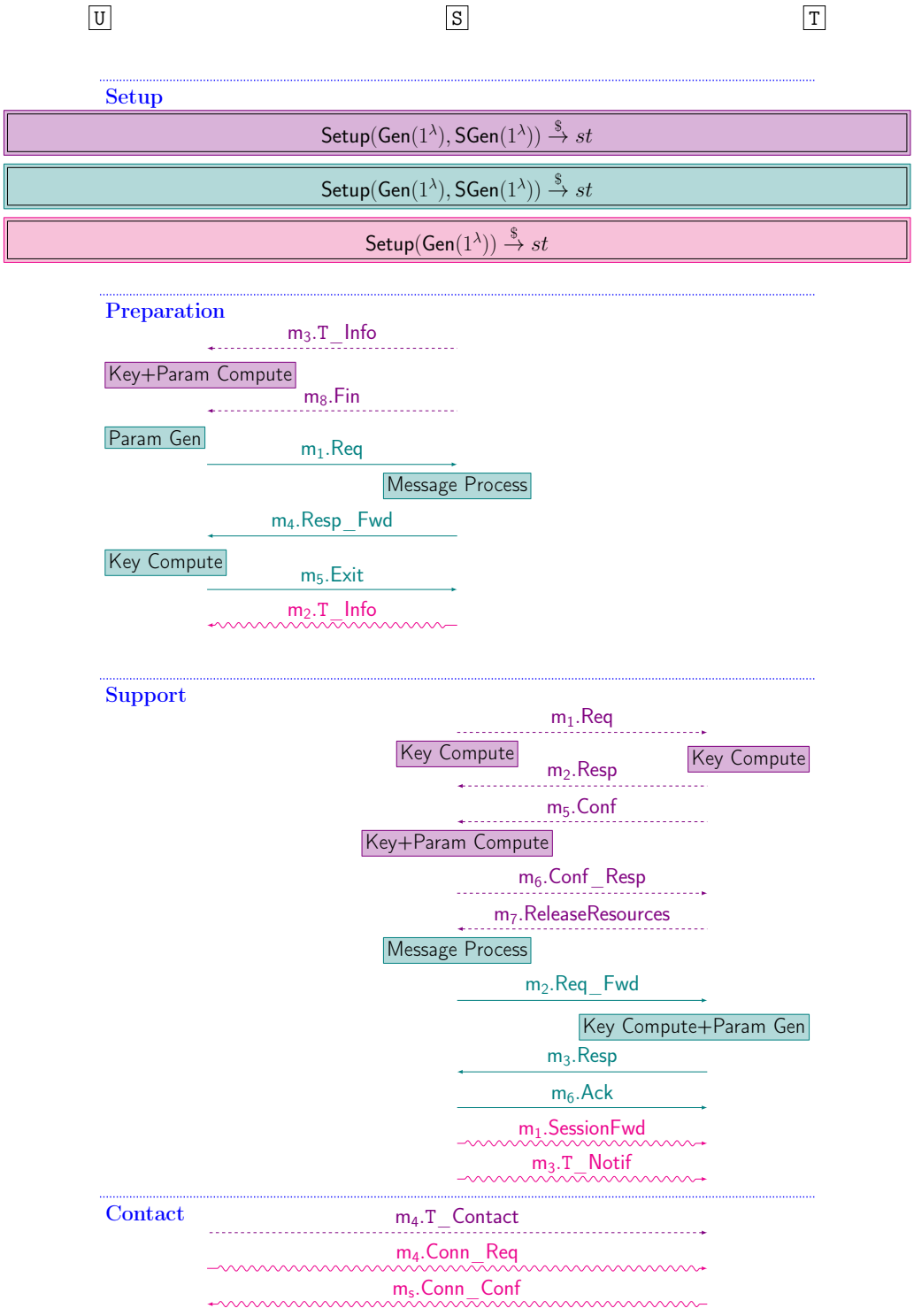


Figure 3.2: Mapping existing HO *schemes to our framework.*
5G-HO | *LDACS*-HO | *CPDLC*-HO

3.2.2 Key Indistinguishability

The majority of secure handover schemes, such as those used by the 5G handover protocol, use secure handover as a mechanism for deriving a shared secret key between the user $\mathtt{U}$ and the target $\mathtt{T}$ by interacting with the source $\mathtt{S}$. This shared secret key can then be used in an arbitrary symmetric key protocol, such as a secure channel protocol, to achieve some secondary goal between the user and target (usually authenticated and confidential communications). Thus, to aid in composability and generalization of our approach, we define *key indistinguishability* of session keys established between $\mathtt{U}$ and $\mathtt{T}$ as the primary goal of secure handover schemes. Since our formalism also produces handover keys, established between $\mathtt{U}$ and $\mathtt{T}$ for future use, our key indistinguishability notion should also cover the security of these handover keys.

Key indistinguishability of secure handover schemes is captured as a game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. $\mathcal{C}$ simulates each user executing a protocol instance, and $\mathcal{A}$ gets to interact with each user. $\mathcal{A}$'s goal is to break key indistinguishability: when a fresh protocol instance has accepted, $\mathcal{A}$ may Test the instance and is given either the real session and handover keys derived in the protocol execution, or random keys sampled uniformly at random from the key distribution space. $\mathcal{A}$'s goal is to determine which keys they have been given and we formalize this goal in Figure 3.3.

Execution Environment

Here we describe the per-session variables maintained by each session instance. Next, we give the explicit definition of security and state $\mathcal{A}$'s advantage in winning this game.

Each session π_i^s maintains the following set of per-session variables:

- $\rho \in \{\mathtt{U}, \mathtt{S}, \mathtt{T}\}$: The role of the party in the current session.
- $i \in \{1, ..., n_P\}$: Index of the session owner.
- $s \in \{1, ..., n_S\}$: Current session index.
- $T_{\mathtt{P}}, T_{\mathtt{S}}, T_{\mathtt{C}}$: Session transcripts of the Prep, Supp and Cont algorithms respectively, initialised by $\perp$.
- $\alpha \in \{\mathtt{prep}, \mathtt{supp}, \mathtt{hand}, \mathtt{accept}, \mathtt{reject}, \perp\}$: The current status of the session, initialized with $\perp$.
- $k \in \{\{0,1\}^\lambda, \perp\}$: Session key to be used in some following symmetric key protocol, or $\perp$ if no session key has yet been computed.

$\underline{\mathsf{Exp}^{\mathsf{KIND,clean}}_{n_S,n_P,\mathcal{A}}(\lambda)}$

1: $b \xleftarrow{\$} \{0,1\}$
2: $\mathtt{tested} \leftarrow (\bot,\bot)$
3: **for** $i = 1$ **to** n_P **do**
4: $\quad pk_i, sk_i \leftarrow \mathsf{Gen}(\lambda)$
5: $\quad \mathbf{ASK}_i \leftarrow \mathtt{false}$
6: $\quad ctr_i \leftarrow 1$
7: $\quad$ **for** $j = 1$ **to** n_S **do**
8: $\quad\quad \mathbf{SSK}_i^j \leftarrow \mathtt{false}$
9: $\quad\quad \mathbf{SK}_i^j \leftarrow \mathtt{false}$
10: $\quad$ **end for**
11: **end for**
12: $b' \leftarrow \mathcal{A}^{\mathbb{Q}}(pk_1, \ldots, pk_{n_P})$
13: $\mathtt{tested} \leftarrow (i^*, s^*)$
14: **if** $(\neg\mathsf{clean}(\pi^{i^*}_{s^*}))$ **then return** $b \xleftarrow{\$} \{0,1\}$
15: **end if**
16: **return** $(b' = b)$

$\underline{\mathsf{Send}(i,s,m)}$

1: **if** $\pi_i^s = \bot$ **then**
2: $\quad$ **return** $\bot$
3: **end if**
4: **if** $\pi_i^s.\alpha = \mathtt{prep}$ **then**
5: $\quad \pi_i^s.st', m' \leftarrow \mathsf{Prep}(\pi_i^s.st, pk_{\pi_i^s.tpid}, m)$
6: **end if**
7: **if** $\pi_i^s.\alpha = \mathtt{supp}$ **then**
8: $\quad \pi_i^s.st', m' \leftarrow \mathsf{Supp}(\pi_i^s.st, pk_{\pi_i^s.\hat{\rho}}, pk_{\pi_i^s.upid}, m)$
9: **end if**
10: **if** $\pi_i^s.\alpha = \mathtt{hand}$ **then**
11: $\quad \pi_i^s.st', m' \leftarrow \mathsf{Cont}(\pi_i^s.st, pk_{\pi_i^s.\hat{\rho}}, pk_{\pi_i^s.spid}, m)$
12: **end if**
13: **return** m'

$\underline{\mathsf{Corrupt}(i)}$

1: $\mathbf{ASK}_i \leftarrow \mathtt{corrupt}$
2: **return** sk_i

$\underline{\mathsf{Compromise}(i,s)}$

1: $\mathbf{SSK}_i^s \leftarrow \mathtt{corrupt}$
2: **return** $\pi_i^s.bk$

$\underline{\mathsf{Reveal}(i,s)}$

1: **if** $\pi_i^s.\alpha \neq \mathtt{accept}$ **then**
2: $\quad$ **return** $\bot$
3: **end if**
4: $\mathbf{SK}_i^s \leftarrow \mathtt{corrupt}$
5: **return** $\pi_i^s.k, \pi_i^s.hk$

$\underline{\mathsf{Create}(i,\rho,j,\ell,s,t)}$

1: **if** $s = \bot$ **then**
2: $\quad s \leftarrow ctr_i$
3: $\quad \pi_i^s.\rho \leftarrow \rho$
4: $\quad \pi_i^s.bk, \pi_i^s.hk, \pi_i^s.k \leftarrow \bot$
5: $\quad$ **if** $(t \neq \bot) \wedge (\pi_i^s.\rho \in \{\mathtt{U},\mathtt{S}\})$ **then**
6: $\quad\quad$ **return** $\bot$
7: $\quad$ **end if**
8: $\quad bk, ad \leftarrow \mathsf{SGen}(\lambda)$
9: $\quad \pi_i^s.bk, \pi_j^t.bk \leftarrow bk$
10: $\quad \pi_i^s.ad, \pi_j^t.ad \leftarrow ad$
11: $\quad$ **if** $(\pi_i^s.\rho = \mathtt{U})$ **then**
12: $\quad\quad \pi_i^s.spid = j, \pi_i^s.tpid = \ell$
13: $\quad$ **end if**
14: $\quad$ **if** $(\pi_i^s.\rho = \mathtt{S})$ **then**
15: $\quad\quad \pi_i^s.upid = j, \pi_i^s.tpid = \ell$
16: $\quad$ **end if**
17: $\quad$ **if** $(\pi_i^s.\rho = \mathtt{T})$ **then**
18: $\quad\quad \pi_i^s.upid = j, \pi_i^s.spid = \ell$
19: $\quad\quad \pi_i^s.bk, \pi_i^s.ad \leftarrow \bot$
20: $\quad$ **end if**
21: **else**
22: $\quad$ **if** $(\pi_i^s.\rho = \mathtt{T})$ **then**
23: $\quad\quad$ **return** $\bot$
24: $\quad$ **end if**
25: $\quad s^* \leftarrow ctr_i$
26: $\quad \pi_i^{s^*}.\rho \leftarrow \rho$
27: $\quad \pi_i^s.hk, \pi_i^s.k \leftarrow \bot$
28: $\quad \pi_i^{s^*}.bk \leftarrow \pi_i^s.hk$
29: $\quad \pi_i^{s^*}.ad \leftarrow \pi_i^s.ad$
30: $\quad s \leftarrow s^*$
31: **end if**
32: $\pi_i^s.T_{\mathtt{P}} \leftarrow \bot$
33: $\pi_i^s.T_{\mathtt{S}} \leftarrow \bot$
34: $\pi_i^s.T_{\mathtt{H}} \leftarrow \bot$
35: $ctr_i{+}{+}$
36: **return** s

$\underline{\mathsf{Test}(i,s)}$

1: **if** $(\pi_i^s.\alpha \neq \mathtt{accept}) \vee (\mathbf{SK}_i^s = \mathtt{corrupt}) \vee ((\bot,\bot) \neq \mathtt{tested})$ **then**
2: $\quad$ **return** $\bot$
3: **end if**
4: $k_0 \xleftarrow{\$} \mathcal{K}, k_1 \leftarrow \pi_i^s.k, \pi_i^s.hk$
5: $\mathtt{tested} \leftarrow (i,s)$
6: **return** k_b

Figure 3.3: The key indistinguishability security experiment for secure HO *schemes. For conciseness we use* $\pi_i^s.\hat{\rho}$ *as shorthand for the communicating partner's party index, i.e. for* Prep $\pi_i^s.\hat{\rho} = \pi_i^s.spid$ *if* $\pi_i^s.\rho = \mathtt{U}$*, and* $\pi_i^s.upid$ *otherwise; for* Supp $\pi_i^s.\hat{\rho} = \pi_i^s.tpid$ *if* $\pi_i^s.\rho = \mathtt{S}$ *and* $\pi_i^s.spid$ *otherwise.* $\mathbb{Q}$ *denotes the set of all queries used in the experiment, i.e.* $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{Test}\}$.

- $hk \in \{\{0,1\}^\lambda, \perp\}$: Handover key to be used as the bootstrap key in some following handover, or $\perp$ if no handover key has yet been computed.
- $bk \in \{\{0,1\}^\lambda, \perp\}$: Bootstrap key used as the initial shared secret between `S` and `U`, or $\perp$ if no bootstrap key has yet been computed. *Note that bootstrap key bk is a result of some initial key-exchange, a previous secure handover between* `U` *and* `S` *or a preshared secret between parties, and is not related to bootstrapping in fully homomorphic encryption.*
- $st \in \{0,1\}^\lambda$: Any additional state used by the session during protocol execution.
- $ad \in \{0,1\}^*$: Some additional data that the `S` advocates to the `T` by the end of the protocol execution.

When the security game is played between the adversary and the challenger, the adversary can issue so-called adversarial queries: this allows the adversary to interact with the challenger's simulated protocol executions. We begin the full list of all adversarial queries below.

- $\mathsf{Create}(i, \rho, j, l, s, t)$: allows the adversary to create a new session π with role ρ owned by party i, with communicating partners j and l. Note that s, t can point to previous sessions π_i^s that has completed and use their output handover key hk as the new bootstrap key bk in the current session. Create also performs some checks to ensure that, if bootstrapping sessions from a previous handover, that it is done consistently, aborting if not.
- $\mathsf{Send}(i, s, m)$: allows the adversary to send the message m to session π_i^s. π_i^s processes m with the appropriate algorithm (i.e. Prep, Supp, or Cont) and returns some output message m' to the adversary.
- $\mathsf{Corrupt}(i)$: allows the adversary to recover the long-term secrets of party i, which enables the framework to capture perfect forward secrecy.
- $\mathsf{Compromise}(i, s)$: allows the adversary to recover the bootstrap key bk used by π_i^s in their protocol execution.
- $\mathsf{Reveal}(i, s)$: allows the adversary to reveal the session key computed by π_i^s in their protocol execution, allowing our model to capture key independence.
- $\mathsf{Test}(i, s)$: returns to the adversary the real-or-random session key and handover keys computed by the so-called *test session* π_i^s, allowing the adversary to play the key indistinguishability game. This query can only be called once.

Cleanness Predicate Much like in key exchange models, our adversary can use the Corrupt, Compromise and Reveal queries to learn secrets. However, with these queries an adversary can trivially impersonate the exposed party to their communicating partner, thus learning the secrets of a potential test session π_i^s. To prevent such trivial attacks, we define *cleanness predicates* that prevent the adversary from making particular patterns of adversarial queries relative to the test session.

It should be noted that such a cleanness predicate is *protocol-specific*, for instance the 5G handover protocol cannot recover from a compromise of the long-term symmetric secret shared between the core network and the user equipment, as opposed to a handover protocol where each party has long-term asymmetric authentication secrets. In Definition 26 we define a cleanness predicate for our StrongHO protocol described in Figure 3.7.

Difficulties in Matching Definitions. Typically, cleanness predicates are used in security experiments to prevent the adversary from issuing adversarial queries that would trivially break the security of the test session. Thus, we need to identify the *matching session* (i.e. the intended communication partner), to determine if the adversary has (for example) Revealed the partner's session key and used it to win the key indistinguishability game.

However, determining the matching partner in a three-party protocol is inherently difficult, especially in a handover protocol where none of the party transcripts are identical. Thus, our model separates party transcripts on a sub-protocol level, i.e. T_{P} for the Prep execution, T_{S} for the Supp execution and T_{C} for the Cont execution, and define matching partners for each sub-protocol. For instance, we say that a session π_i^s UT-matches a partner session π_j^t if $\pi_i^s.\rho \neq \pi_j^t.\rho$ and $\pi_i^s.\rho, \pi_j^t.\rho \in \{\mathtt{U}, \mathtt{T}\}$ and $\pi_i^s.T_{\mathsf{C}} = \pi_j^t.T_{\mathsf{C}}$.

Definition 26 (Strong Handover KIND cleanness predicate)**.** *A session π_i^s such that $\pi_i^s.\alpha =$ accept in the security experiment defined in Figure 3.3 is clean (i.e., $\mathsf{clean}_{\mathsf{str\text{-}kind}}(\pi_i^s) = 1$) if all of the following conditions hold:*

1. $\mathbf{SK}_i^s \neq$ corrupt ***(Session key has not been exposed. This condition must hold true for all secure*** HO ***constructions)****;*
2. *For all $(j,t) \in n_P \times n_S$ such that π_i^s UT-matches π_j^t,* $\mathbf{SK}_j^t \neq$ corrupt ***(Session key not exposed at partner session. This condition must hold true for all secure*** HO ***constructions)****;*

3. *If* $\pi_i^s.\rho = \texttt{U}$*, and there exists no session* π_j^t *that* UT*-matches* π_i^s*, and there exists a session* π_l^r *such that* $\pi_i^s.bk = \pi_l^r.bk$*, then* $\mathbf{SSK}_i^s \neq \texttt{corrupt}$ *nor* $\mathbf{SSK}_l^r \neq \texttt{corrupt}$ ***(If the user session has no matching*** UT ***partner, then their bootstrap key has not been exposed)****;*
4. *If* $\pi_i^s.\rho = \texttt{U}$*, and there exists no session* π_j^t *that* UT*-matches* π_i^s*, and there exists a session* π_l^r *such that* $\pi_i^s.bk = \pi_l^r.hk$*, then* $\mathbf{SK}_l^r \neq \texttt{corrupt}$ ***(If the user session has no matching*** UT ***partner, then their previous handover key has not been exposed)****;*
5. *If* $\pi_i^s.\rho = \texttt{T}$*, and there exists no session* π_j^t *that* UT*-matches* π_i^s*, but there exists a session* π_l^r *such that* $\pi_l^r.i = \pi_i^s.upid$ *or* $\pi_l^r.upid = \pi_i^s.upid$*, and* $\pi_l^r.bk \neq \perp$*, then* $\mathbf{SK}_l^r \neq \texttt{corrupt}$ ***(If the target session has no matching*** UT ***partner, then the user partner's bootstrap key has not been exposed)****;*
6. *If* $\pi_i^s.\rho = \texttt{T}$*, and there exists no session* π_j^t *that* UT*-matches* π_i^s*, but there exists a session* π_l^r *such that* $\pi_l^r.i = \pi_i^s.upid$ *or* $\pi_l^r.upid = \pi_i^s.upid$*, and* $\pi_l^r.hk \neq \perp$*, then* $\mathbf{SK}_l^r \neq \texttt{corrupt}$ ***(If the target session has no matching*** UT ***partner, then the user partner's previous handover key has not been exposed)****;*
7. *If there exists no session* π_j^t *that* UT*-matches* π_i^s*, then* $\mathbf{ASK}_i \neq \texttt{corrupt} \forall i$ ***(If the test session has no matching*** UT ***partner, then no source long-term key has been exposed)****;*
8. *If there exists no session* π_j^t *that* UT*-matches* π_i^s*, and* $\pi_i^s.\rho = \texttt{U}$ *then* $\mathbf{ASK}_{\pi_i^s.tpid} \neq \texttt{corrupt}$ ***(If the user session has no matching*** UT ***partner, then the target long-term key has not been exposed)****;*

Broadly speaking, this captures a perfect forward secret handover scheme - note that the adversary is allowed to compromise the long-term secrets of any party participating in the protocol execution after the test session has completed.

We now turn to defining formally the key indistinguishability of secure handover schemes.

Definition 27 (KIND Key Indistinguishability)**.** *Let* HO *be a secure handover protocol, and* n_P*,* $n_S \in \mathbb{N}$*. For a particular given predicate* clean*, and a PPT algorithm* $\mathcal{A}$*, we define the advantage of* $\mathcal{A}$ *in the* KIND *key indistinguishability game to be:* $\mathsf{Adv}^{\mathsf{KIND,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{KIND,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$*. We say that* HO *is* KIND*-secure if, for all* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{KIND,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

3.2.3 Unlinkability

Another important security property that is often discussed in the context of secure handover schemes is *user anonymity*. For example, the 5G-HO introduced identity-hiding techniques in order to prevent attackers from learning the identity of the user communicating with the 5G network. We note that, while *anonymity* and *unlinkablity* are often used interchangeably, we define them as distinct security properties. We argue that *unlinkablity* offers a stronger guarantee, inherently preserving user *anonymity*. For example, if an adversary links two sessions to a single pseudo-identity, *anonymity* is preserved but *unlinkablity* is compromised. However, maintaining *unlinkablity* ensures *anonymity*, as it prevents an adversary from linking sessions to an individual user. Thus, we treat *unlinkablity* and *anonymity* as distinct properties, with *unlinkablity* providing stronger security guarantees. To capture this, we formalize the notion of *user unlinkability* (UNLINK). Much like KIND, the UNLINK property is captured as a game played between $\mathcal{A}$ and $\mathcal{C}$ where $\mathcal{A}$ is meant to guess some bit b sampled by $\mathcal{C}$.

However, unlike KIND, the UNLINK game allows the adversary to specify two (distinct) U parties, and the $\mathcal{C}$ uses the random bit b to determine *which* user will run the so-called Test session that $\mathcal{A}$ will be interacting with. We note that this adversary $\mathcal{A}$ is an external attacker, who tries to *link* sessions to a specific user U. Since for all other protocol executions $\mathcal{A}$ is allowed to specify the user executing the protocol, then by *linking* two protocol sessions run by the same user, $\mathcal{A}$ will be able to determine the identity of the Test session and thus the bit b. We formalize this goal in Figure 3.4.

Similar to key indistinguishability described in Section 3.2.2, when the security game is played between $\mathcal{A}$ and $\mathcal{C}$, $\mathcal{A}$ can issue so-called adversarial queries, allowing $\mathcal{A}$ to interact with $\mathcal{C}$'s simulated protocol executions. The TestUnlink and SendTest queries replace Test from the KIND experiment which we present below. All other queries remain identical.

– TestUnlink$((i, s), (i', s'), j, (t, t'), l)$: allows the adversary to create the Test session π_b and its S and T partners. The adversary is able to specify two party identifiers i, i' that the challenger will create a single protocol execution for, and the adversary's goal is to distinguish which party (i or i') owns π_b. The majority of operations in TestUnlink is administrative management to ensure that the adversary can point to previous sessions (and thus use a previously computed handover key hk), but not trivially break the UNLINK security of the protocol. Note that j and ℓ are communicating party identifiers while s, s', t and, t' identify sessions.

$\underline{\mathsf{Exp}^{\mathsf{UNLINK},\mathsf{clean}}_{n_S,n_P,\mathcal{A}}(\lambda)}$

1: $b \xleftarrow{\$} \{0,1\}$
2: **for** $i = 1$ **to** n_P **do**
3: $\quad pk_i, sk_i \leftarrow \mathsf{Gen}(\lambda)$
4: $\quad \mathbf{ASK}_i \leftarrow \mathtt{false}$
5: $\quad ctr_i \leftarrow 1$
6: $\quad$ **for** $j = 1$ **to** n_S **do**
7: $\quad\quad \mathbf{SSK}_i^j \leftarrow \mathtt{false}$
8: $\quad\quad \mathbf{SK}_i^j \leftarrow \mathtt{false}$
9: $\quad$ **end for**
10: **end for**
11: $b' \leftarrow \mathcal{A}^{\mathbb{Q}}(pk_1, \ldots, pk_{n_P})$
12: **if** $(\neg\mathsf{clean}(\pi_b) \vee \neg\mathsf{clean}(\pi_{b-1}))$ **then**
13: $\quad$ **return** $b \xleftarrow{\$} \{0,1\}$
14: **else**
15: $\quad$ **return** $(b' = b)$
16: **end if**

$\underline{\mathsf{SendTest}(m)}$

1: $\mathrm{Send}(\pi_b, m) \rightarrow m'$
2: **return** m'

$\underline{\mathsf{TestUnlink}((i,s),(i',s'),j,(t,t'),\ell)}$

1: **if** $((s \neq \perp) \wedge (s' = \perp)) \vee ((s = \perp) \wedge (s' \neq \perp))$ **then**
2: $\quad$ **return** $\perp$
3: **end if**
4: **if** $((t \neq \perp) \wedge (t' = \perp)) \vee ((t = \perp) \wedge (t' \neq \perp))$ **then**
5: $\quad$ **return** $\perp$
6: **end if**
7: **if** $(\mathbf{SK}_i^s = \mathtt{corrupt}) \vee (\mathbf{SK}_{i'}^{s'} = \mathtt{corrupt}) \vee (\mathbf{SK}_j^t = \mathtt{corrupt}) \vee (\mathbf{SK}_{j'}^{t'} = \mathtt{corrupt})$ **then**
8: $\quad$ **return** $\perp$
9: **end if**
10: $s \leftarrow \mathsf{Create}(i, \mathtt{U}, j, \ell, s)$
11: $s' \leftarrow \mathsf{Create}(i', \mathtt{U}, j, \ell, s')$
12: $t \leftarrow \mathsf{Create}(j, \mathtt{S}, i, \ell, t)$
13: $t' \leftarrow \mathsf{Create}(j, \mathtt{S}, i', \ell, t')$
14: **if** $(s = \perp) \vee (s' = \perp) \vee (t = \perp) \vee (t' = \perp)$ **then**
15: $\quad$ **return** $\perp$
16: **end if**
17: **if** $b = 0$ **then**
18: $\quad \pi_b \leftarrow \pi_i^s$
19: $\quad \pi_{b-1} \leftarrow \pi_{i'}^{s'}$
20: $\quad r \leftarrow \mathsf{Create}(\ell, \mathtt{T}, i, j, \perp)$
21: $\quad \pi_i^s, \pi_{i'}^{s'}, \pi_j^{t'} \leftarrow \perp$
22: $\quad ctr_i \leftarrow ctr_i - 1, ctr_{i'} \leftarrow ctr_{i'} - 1$
23: $\quad ctr_j \leftarrow ctr_j - 1$
24: **else**
25: $\quad \pi_{b-1} \leftarrow \pi_i^s$
26: $\quad \pi_b \leftarrow \pi_{i'}^{s'}$
27: $\quad r \leftarrow \mathsf{Create}(\ell, \mathtt{T}, i', j, \perp)$
28: $\quad \pi_j^t \leftarrow \pi_j^{t'}$
29: $\quad \pi_i^s, \pi_{i'}^{s'}, \pi_j^{t'} \leftarrow \perp$
30: $\quad ctr_i \leftarrow ctr_i - 1, ctr_{i'} \leftarrow ctr_{i'} - 1$
31: $\quad ctr_j \leftarrow ctr_j - 1$
32: **end if**
33: **return** (t, r)

Figure 3.4: The unlinkability security experiment for secure HO *schemes. For conciseness we only give the definition of the overall experiment, the* SendTest *and* TestUnlink *queries, as all other adversarial queries are identical to the* KIND *experiment described in Figure 3.3.* $\mathbb{Q}$ *denotes the set of all queries used in the experiment, i.e.* $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{TestUnlink}, \mathsf{SendTest}\}$. $\mathcal{C}$ *also maintains a counter ctr to ensure that the changes made to the two sessions are consistent and* $\mathcal{A}$ *cannot gain any additional information about the* Test *session.*

– $\mathsf{SendTest}(m)$: allows the adversary to send a message m to the Test session π_b.

Definition 28 (Strong Handover UNLINK cleanness predicate)**.** *A session* π_i^s *such that*

$\pi_i^s.\alpha = \texttt{accept}$ *in the security experiment defined in Figure 3.4 is* $\mathsf{clean}_{\mathsf{str\text{-}unlink}}$ *if all of the following conditions hold:*

1. $\mathsf{clean}_{\mathsf{str\text{-}kind}}(\pi_i^s) = 1$*;* ***(The session*** π_i^s ***is clean as defined in Definition 26)***
2. *If there exists some session* $\pi_{i'}^{s'}$ *such that* $\pi_i^s.bk = \pi_{i'}^{s'}.hk$*, then* $\mathsf{clean}_{\mathsf{str\text{-}kind}}(\pi_i^{s'}) = 1$*;* ***(Any previous handover session*** $\pi_{i'}^{s'}$ ***is clean as defined in Definition 26)***

We note here that it is trivial to link a test session π_b in the UNLINK security experiment simply by the adversary using some previous session $\pi_i^{s'}$ to generate the bootstrap key bk for π_b, and then simply Reveal-ing the hk from $\pi_i^{s'}$ and later Compromise-ing π_b. To prevent such an attack, we require that the test session π_b *and* the previous session $\pi_i^{s'}$ π_b that is bootstrapped, are both clean according to Definition 28.

We build the cleanness predicate for UNLINK on our KIND-clean of Definition 26. This draws from how a user U in StrongHO is linked to previous and future sessions by bootstrap key bk and handover key hk, respectively. Establishing that these keys are uniformly random and independent and thus cannot be exploited to link user U to specific sessions forms the foundation of UNLINK security within our construction.

We give the explicit definition of security below and state $\mathcal{A}$'s advantage in winning this game in Definition 29.

Definition 29 (UNLINK Unlinkability)**.** *Let* HO *be a secure handover protocol, and* n_P*,* $n_S \in \mathbb{N}$*. For a particular given predicate* clean*, and a PPT algorithm* $\mathcal{A}$*, we define the advantage of* $\mathcal{A}$ *in the* UNLINK *unlinkability game to be:* $\mathsf{Adv}^{\mathsf{UNLINK,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{UNLINK,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$*. We say that* HO *is* UNLINK*-secure if, for all* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{UNLINK,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

3.2.4 Target and Source Privacy

In some handover schemes, knowing the path that the user takes (i.e. the paths between different source and target nodes that the user transitions between) can be private information that is worth protecting. For instance, during a CPDLC handover, identifying the air traffic control station that a covert military aircraft communicates with would enable an attacker to recover their general geographical location. Or, consider how the existing 5G handover implementation allows the source S to communicate directly with the target T [97], thereby revealing the exact trajectory of the user U's movement across the nodes involved. In fact, the work of Miller et al. [91] recognize the feasibility of such insider threat

vectors, and extend [97] to model these adversarial nodes. We argue that in this context, the risk of rogue nodes conducting unauthorized tracking introduces a new insider threat which has not yet been sufficiently addressed within existing academic literature or any real-world implementations.

To formalize the security of this information in our framework, we introduce Target and Source Privacy (denoted as TPRIV and SPRIV respectively) which we collectively identify as *path privacy* PPRIV. On a high-level, TPRIV and SPRIV respectively prevent an insider source (resp. target) from learning which target (resp. source) the user was communicating with. Much like KIND, the TPRIV (resp. SPRIV) property is captured as a game played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ where $\mathcal{A}$ is meant to guess some bit b sampled by $\mathcal{C}$. Much like the UNLINK game, the TPRIV (resp. SPRIV) adversary selects two distinct T (resp. source) parties, and the challenger uses the random bit b sampled to determine which target (resp. source) the Test session that $\mathcal{A}$ will be interacting with.

Unlike the UNLINK game, the threat model considered here is an *insider* attacker: in TPRIV the adversary is allowed to Compromise the source party that the user and target will interact with (and in SPRIV, the target party that the user and source will interact with). This will allow a secure handover scheme to argue for *path privacy*: SPRIV ensures that target nodes do not know which source node the user came from, and TPRIV ensures that the source nodes do not know which target node the user went to.

We formalize this game in Figure 3.5. We give the explicit definition of security below and state $\mathcal{A}$'s advantage in winning this game.

Definition 30 (Strong Handover TPRIV cleanness predicate)**.** *A session π_i^s such that $\pi_i^s.\alpha =$ accept in the security experiment defined in Figure 3.5 is* $\mathsf{clean}_{\mathsf{str\text{-}tpriv}}$ *if all of the following conditions hold:*

1. $\mathbf{SK}_i^s \neq$ corrupt ***(Session key has not been exposed);***
2. *For all $(j,t) \in n_P \times n_S$ such that π_i^s UT-matches π_j^t, $\mathbf{SK}_j^t \neq$ corrupt **(Session key not exposed at partner session);***
3. $\mathbf{ASK}_i \neq$ corrupt ***(Target's long-term key has not been exposed);***

We note here that it is trivial to link a test session π_b in the TPRIV security experiment to some future session π by using π_b to generate the bootstrap key bk for π, and then simply Compromise-ing the bk from π. To prevent such an attack, we require that the test session π_b is itself KIND-secure. KIND-security of π_b guarantees that all subsequent

$\underline{\mathsf{Exp}^{\mathsf{TPRIV,clean}}_{n_S,n_P,\mathcal{A}}(\lambda)}$

1: $b \xleftarrow{\$} \{0,1\}$
2: **for** $i = 1$ **to** n_P **do**
3: $\quad pk_i, sk_i \leftarrow \mathsf{Gen}$
4: $\quad \mathbf{ASK}_i \leftarrow \texttt{false}$
5: $\quad ctr_i \leftarrow 1$
6: $\quad$ **for** $j = 1$ **to** n_S **do**
7: $\quad\quad \mathbf{SSK}^j_i \leftarrow \texttt{false}$
8: $\quad\quad \mathbf{SK}^j_i \leftarrow \texttt{false}$
9: $\quad$ **end for**
10: **end for**
11: $b' \leftarrow \mathcal{A}^{\mathbb{Q}}(pk_1, \ldots, pk_{n_P})$
12: **if** $(\neg\mathsf{clean}(\pi_b) \lor \neg\mathsf{clean}(\pi_{b-1}))$ **then**
13: $\quad$ **return** $b \xleftarrow{\$} \{0,1\}$
14: **else**
15: $\quad$ **return** $(b' = b)$
16: **end if**

$\underline{\mathsf{SendTest}(m)}$

1: $\mathsf{Send}(\pi_b, m) \to m'$
2: **return** m'

$\underline{\mathsf{TestTarget}(i,s),(j,t),\ell,\ell'}$

1: **if** $(\mathbf{ASK}_\ell = \texttt{corrupt}) \lor (\mathbf{ASK}_{\ell'} = \texttt{corrupt})$ **then**
2: $\quad$ **return** $\perp$
3: **end if**
4: **if** $(b = 0)$ **then**
5: $\quad \ell^* \leftarrow \ell$
6: **else**
7: $\quad \ell^* \leftarrow \ell'$
8: **end if**
9: $s' \leftarrow \mathsf{Create}(i, \mathtt{U}, j, \ell^*, s)$
10: $t' \leftarrow \mathsf{Create}(j, \mathtt{S}, i, \ell^*, t)$
11: **if** $(s' = \perp) \lor (t' = \perp)$ **then**
12: $\quad$ **return** $\perp$
13: **end if**
14: $r \leftarrow \mathsf{Create}(\ell^*, \mathtt{T}, i, j, \perp)$
15: $\pi_b \leftarrow \pi^r_{\ell^*}, \pi^r_{\ell^*} \leftarrow \perp, ctr_{\ell^*} \leftarrow ctr_{\ell^*} - 1$
16: **return** (t, r)

Figure 3.5: The target privacy security experiment for secure HO *schemes. For conciseness we only give the definition of the overall experiment, the* SendTest *and* TestTarget *queries, as all other adversarial queries are identical to the* KIND *experiment described in Figure 3.3.* $\mathbb{Q}$ *denotes the set of all queries used in the experiment, i.e.* $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{TestTarget}, \mathsf{SendTest}\}$. $\mathcal{C}$ *also maintains a counter ctr to ensure that the changes made to the two sessions are consistent and* $\mathcal{A}$ *cannot gain any additional information about the* Test *session.*

keys derived from bk that encrypt communications between user U and source S remain uniformly random and independent. Note that since target sessions do not bootstrap from some previous session, we do not require any previous session π be KIND-secure. Finally, since the target uses their long-term PKE key to decrypt ciphertexts from the user session, we cannot allow the adversary to Corrupt it. Next, we formalize TPRIV security in the following Definition 31.

Definition 31 (Target Privacy Security for Handover Schemes)**.** *Let* HO *be a secure handover protocol, and* n_P, $n_S \in \mathbb{N}$. *For a particular given predicate* clean*, and a PPT algorithm* $\mathcal{A}$*, we define the advantage of* $\mathcal{A}$ *in the* TPRIV *game to be:* $\mathsf{Adv}^{\mathsf{TPRIV,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) =$

$|\Pr[\mathsf{Exp}^{\mathsf{TPRIV},\mathsf{clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)=1]-\frac{1}{2}|$.

We say that HO *is* TPRIV*-secure if, for all* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{TPRIV},\mathsf{clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ.

For completeness, next we give a similar definition of security for source privacy along with cleanness predicate for SPRIV. We also give a formalization of the source privacy game in Figure 3.6.

Definition 32 (Strong Handover SPRIV cleanness predicate)**.** *A session* π_i^s *such that* $\pi_i^s.\alpha = \texttt{accept}$ *in the security experiment defined in Figure 3.6 is* $\mathsf{clean}_{\mathsf{str\text{-}spriv}}$ *if all of the following conditions hold:*

1. $\mathbf{SK}_i^s \neq \texttt{corrupt}$ ***(Session key has not been exposed)****;*
2. *For all* $(j,t) \in n_P \times n_S$ *such that* π_i^s US*-matches* π_j^t, $\mathbf{SK}_j^t \neq \texttt{corrupt}$ ***(Session key not exposed at partner session)****;*
3. $\mathbf{ASK}_i \neq \texttt{corrupt}$ ***(The source long-term key has not been exposed)****;*
4. *If there exists a session* π_j^t *such that* $\pi_i^s.bk = \pi_j^t.bk$*, then* $\mathbf{SSK}_j^t \neq \texttt{corrupt}$ ***(Any matching bootstrap key has not been exposed)****;*
5. *If there exists a session* π_j^t *such that* $\pi_i^s.bk = \pi_j^t.hk$*, then* $\mathbf{SK}_j^t \neq \texttt{corrupt}$ ***(Any matching handover key has not been exposed)****;*
6. *If there exists a session* π_j^t *such that* $\pi_i^s.bk = \pi_j^t.hk$*, then* $\mathsf{clean}_{\mathsf{str\text{-}kind}}(\pi_j^t)$ ***(Any previous handover session has derived good handover keys)****;*

Unlike the target T in TPRIV, source S in SPRIV already has an existing session π with user U, which maybe an initial session or a post-handover session where the current S was the T in a previous handover. We note here that it is trivial to link a test session π_b in the SPRIV security experiment simply by the adversary using some previous session π to generate the bootstrap key bk for π_b, and then simply Reveal-ing the hk from π. To prevent such an attack, we require both that the test session π_b is itself KIND-secure, and also that any previous session π that π_b is bootstrapped from is also KIND-secure. Finally, since the source uses their long-term PKE key to decrypt ciphertexts from the user session, we cannot allow the adversary to Corrupt it. Next, we formalize SPRIV security in the following Definition 33.

Definition 33 (Source Privacy Security for Handover Schemes)**.** *Let* HO *be a secure handover protocol, and* n_P, $n_S \in \mathbb{N}$. *For a particular given predicate* clean,

and a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the SPRIV *game to be:* $\mathsf{Adv}^{\mathsf{SPRIV,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{SPRIV,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$. *We say that* HO *is* SPRIV*-secure if, for all $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{SPRIV,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

We finish our definitions for HO security by giving a formal definition for path privacy, which encapsulates both SPRIV and TPRIV. We note that while our definition is upper-bounded by the advantages of SPRIV and TPRIV, within our construction, breaking either SPRIV or TPRIV security indicates a successful attack on PPRIV security.

Definition 34 (Path Privacy for Handover Schemes)**.** *Let* HO *be a secure handover protocol, and n_P, $n_S \in \mathbb{N}$. For the advantages for* TPRIV *(Definition 31) and* SPRIV *(Definition 33), and a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ against path privacy to be:* $\mathsf{Adv}^{\mathsf{PPRIV}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{TPRIV,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{SPRIV,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$. *We say that* HO *is* PPRIV*-secure if, for all $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{PPRIV}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

3.3 Strong Handover Scheme

In this section, we construct a generic strong handover protocol that capture all notions of security that we describe in Section 3.2. Our StrongHO strong handover construction captures all previously formalized security notions for handovers (KIND, UNLINK), as well as our novel path privacy notions (SPRIV, TPRIV). On a high-level, the user `U` generates a ciphertext (encrypting a symmetric key under the target `T`'s public key) and a KEM public key. Both are passed to the source `S` for authentication, which signs and MACs both values. The user verifies the signature, and deletes it from the message, sending the KEM public key, the ciphertext and the MAC tag to the target. The target verifies the MAC tag and is satisfied that both values come from some authenticated user. The target encapsulates a fresh secret under the user's public key, and both parties derive the same set of keys.

We leverage a shared symmetric authentication key *ak* generated by a puncturable PRF between all source and target nodes to facilitate the *path privacy* of `U` on the move. We note that in order to guarantee *path privacy* within StrongHO, we omit the Supp phase which inherently precludes any notion of privacy between source `S` and target `T`. The PPRF fortifies the security of our construction by rendering the current *ak* unusable, after it anonymously authenticates `S` to `T` for the ongoing HO session. The use of PPRF in this manner prevents replay attacks, and guarantees forward secrecy. StrongHO has only three phases, which are described below and illustrated in Figure 3.7.

$\underline{\mathsf{Exp}^{\mathsf{SPRIV,clean}}_{n_S,n_P,\mathcal{A}}(\lambda)}$

1: $b \xleftarrow{\$} \{0,1\}$
2: **for** $i = 1$ **to** n_P **do**
3: $\quad pk_i, sk_i \leftarrow \mathsf{Gen}$
4: $\quad \mathbf{ASK}_i \leftarrow \mathtt{false}$
5: $\quad ctr_i \leftarrow 1$
6: $\quad$ **for** $j = 1$ **to** n_S **do**
7: $\quad\quad \mathbf{SSK}^j_i \leftarrow \mathtt{false}$
8: $\quad\quad \mathbf{SK}^j_i \leftarrow \mathtt{false}$
9: $\quad$ **end for**
10: **end for**
11: $b' \leftarrow \mathcal{A}^{\mathbb{Q}}(pk_1, \ldots, pk_{n_P})$
12: **if** $(\neg\mathsf{clean}(\pi_b) \lor \neg\mathsf{clean}(\pi_{b-1}))$ **then**
13: $\quad$ **return** $b \xleftarrow{\$} \{0,1\}$
14: **else**
15: $\quad$ **return** $(b' = b)$
16: **end if**

$\underline{\mathsf{SendTest}(m)}$

1: $\mathrm{Send}(\pi_b, m) \rightarrow m'$
2: **return** m'

$\underline{\mathsf{TestSource}((i,s,s'),(j,t),(j',t'),\ell)}$

1: **if** $((s \neq \bot) \land (s' = \bot)) \lor ((s = \bot) \land (s' \neq \bot))$ **then**
2: $\quad$ **return** $\bot$
3: **end if**
4: **if** $((t \neq \bot) \land (t' = \bot)) \lor ((t = \bot) \land (t' \neq \bot))$ **then**
5: $\quad$ **return** $\bot$
6: **end if**
7: **if** $(\mathbf{SK}^s_i = \mathtt{corrupt}) \lor (\mathbf{SK}^{s'}_i = \mathtt{corrupt}) \lor (\mathbf{SK}^t_j = \mathtt{corrupt}) \lor (\mathbf{SK}^{t'}_{j'} = \mathtt{corrupt})$ **then**
8: $\quad$ **return** $\bot$
9: **end if**
10: $s \leftarrow \mathsf{Create}(i, \mathtt{U}, j, \ell, s)$
11: $s' \leftarrow \mathsf{Create}(i, \mathtt{U}, j', \ell, s')$
12: $t \leftarrow \mathsf{Create}(j, \mathtt{S}, i, \ell, t)$
13: $t' \leftarrow \mathsf{Create}(j', \mathtt{S}, i, \ell, t')$
14: **if** $(s = \bot) \lor (s' = \bot) \lor (t = \bot) \lor (t' = \bot)$ **then**
15: $\quad$ **return** $\bot$
16: **end if**
17: **if** $b = 0$ **then**
18: $\quad \pi_b \leftarrow \pi^t_j$
19: $\quad \pi_{b-1} \leftarrow \pi^{t'}_{j'}$
20: $\quad r \leftarrow \mathsf{Create}(\ell, \mathtt{T}, i, j, \bot)$
21: $\quad \pi^{s'}_i \pi^t_j, \pi^{t'}_{j'} \leftarrow \bot$
22: **else**
23: $\quad \pi_{b-1} \leftarrow \pi^t_j$
24: $\quad \pi_b \leftarrow \pi^{t'}_{j'}$
25: $\quad r \leftarrow \mathsf{Create}(\ell, \mathtt{T}, i, j', \bot)$
26: $\quad \pi^s_i \leftarrow \pi^{s'}_i$
27: $\quad \pi^{s'}_i \pi^t_j, \pi^{t'}_{j'} \leftarrow \bot$
28: **end if**
29: **return** (s, r)

Figure 3.6: The source privacy security experiment for secure HO *schemes. For conciseness we only give the definition of the overall experiment, the* SendTest *and* TestSource *queries, as all other adversarial queries are identical to the* KIND *experiment described in Figure 3.3.* $\mathbb{Q}$ *denotes the set of all queries used in the experiment, i.e.* $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{TestSource}, \mathsf{SendTest}\}$. $\mathcal{C}$ *also maintains a counter ctr to ensure that the changes made to the two sessions are consistent and* $\mathcal{A}$ *cannot gain any additional information about the* Test *session.*

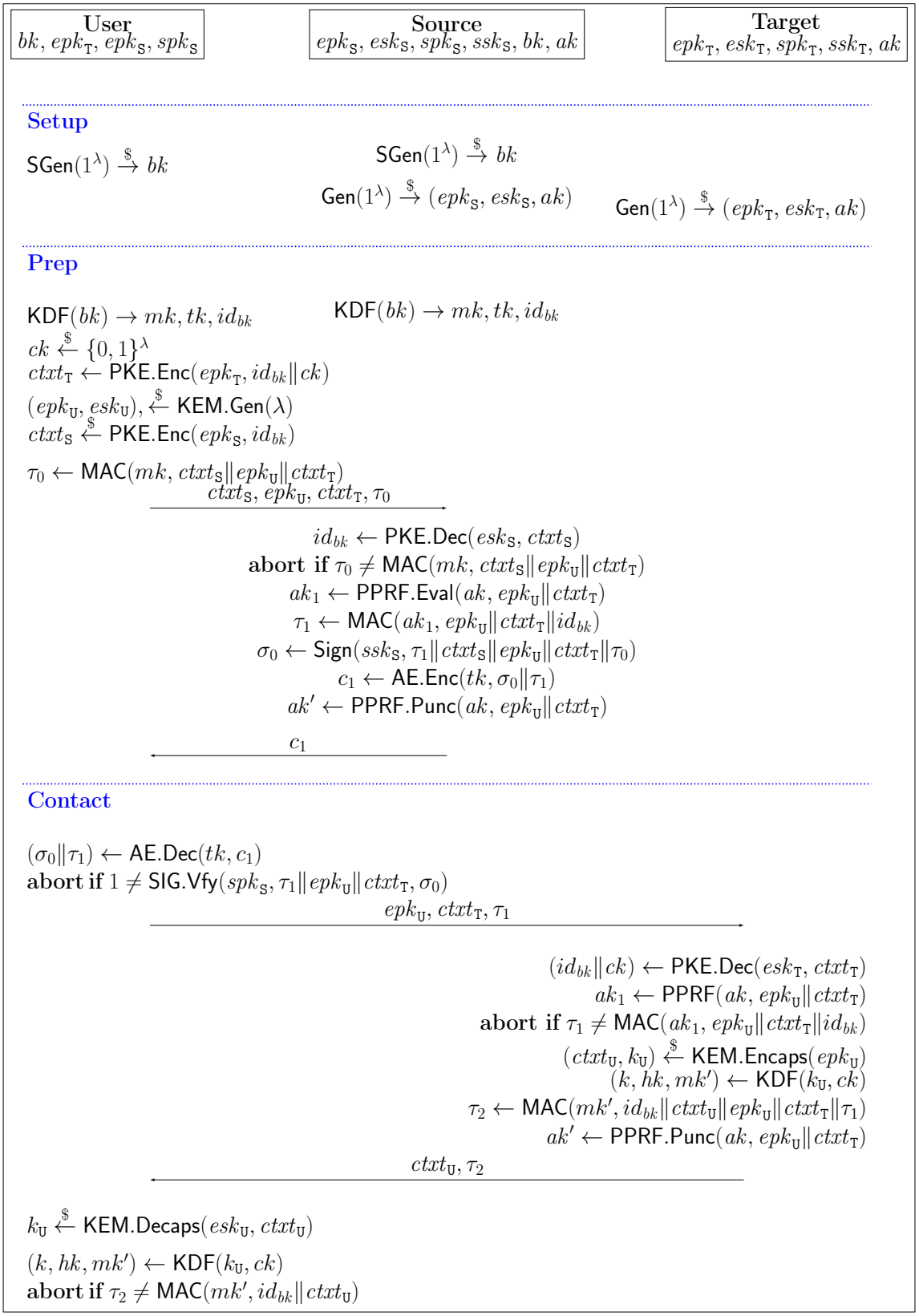


Figure 3.7: The StrongHO *protocol.*

Setup : During this phase long-term asymmetric key pairs, long-term symmetric secrets and ephemeral bootstrap secrets are established.

Preparation : During this phase, U communicates a ephemeral KEM public key and a PKE ciphertext for S to authenticate T. The phase starts with U deriving new keys and identifiers mk, tk, id_{bk} from bk. U samples a random key ck to be communicated to T, in order to introduce additional entropy into session keys derived between U and T (and thus, preventing S from also deriving them). This key is encrypted along with id_{bk} using T's long-term encryption key $epk_{\mathtt{T}}$, generating $ctxt_{\mathtt{T}}$. We note that the encryption of $ctxt_{\mathtt{T}}$ under PKE is essential to ensure ciphertext anonymity, thereby preserving the TPRIV property of the target T. U generates a new ephemeral KEM key-pair $epk_{\mathtt{U}}, esk_{\mathtt{U}}$. This ensures the key-indistinguishability of the session keys generated, i.e. achieving *perfect forward secrecy*. Next, U encrypts id_{bk} under the long-term encryption key of S $epk_{\mathtt{S}}$. Finally, U generates a MAC tag τ_0 on both ciphertexts along with $epk_{\mathtt{U}}$ and sends them to S. After receiving the message, S decrypts $ctxt_{\mathtt{S}}$ to obtain id_{bk}, identifying the correct bk. Upon successful verification of τ_0, S extracts a PPRF authentication key ak' for the session, using the master authentication key ak evaluated over $epk_{\mathtt{U}}$ and $ctxt_{\mathtt{T}}$. S then calculates a MAC tag τ_1 with ak' over $epk_{\mathtt{U}}$ and $ctxt_{\mathtt{T}}$, which will be verified by T during the Contact phase. Next S generates σ_0 by signing τ_1, $epk_{\mathtt{U}}$ and $ctxt_{\mathtt{T}}$ with its long-term signing key $ssk_{\mathtt{S}}$ which is then encrypted under tk along with τ_1 to produce c_1. The encryption of σ_0 prevents an adversary from identifying S's identity, and thereby breaking source privacy SPRIV, via the publicly available long-term signing key of S. Finally, S punctures the master authentication key ak for $epk_{\mathtt{U}}$ and $ctxt_{\mathtt{T}}$ and returns c_1 to U.

Contact : At the beginning of this stage U decrypts and verify σ_0. Upon successful verification, U and T proceed to authenticate each other and establish a shared secret state. U initiates the authentication process by sending $ctxt_{\mathtt{T}}, pk_{\mathtt{U}}$ and the S-generated MAC tag τ_1 to T. Following reception, T decrypts $ctxt_{\mathtt{T}}$ to obtain id_{bk}, ck and identify U via id_{bk}, which T stores as a session identifier for its future UT sessions. T then verifies MAC tag τ_1 with ak' and, carries on to encapsulate U's public key $epk_{\mathtt{U}}$ to derive $k_{\mathtt{U}}$ and $ctxt_{\mathtt{U}}$. Next, T uses the newly derived key $k_{\mathtt{U}}$ and decrypted ck to generate a set of keys (k, hk, mk'). Using mk', T generates a MAC tag τ_2 for $(id_{bk}, ctxt_{\mathtt{U}})$ and finally sends them back to U.

Upon receiving the message, U decapsulates $ctxt_{\mathtt{U}}$ and derives a set of shared keys for the new session. The Handover completes once U successfully verifies the MAC tag τ_2.

Remark 1. *One observation is that typically, when the efficiency of round-trip time is prioritized, there exists an inherent trade-off between the path privacy of a secure* HO *protocol and its compromise resilience. Consider a secure* HO *scheme that achieves source and target privacy solely via a single shared group key ak. All nodes, targets and sources, share this single symmetric ak for authenticating user secrets. It's clear that this* HO *achieves source privacy, since the authentication token could have come from any source node. Similarly, the* HO *achieves target privacy, since this token validly authenticates to any target node. However, this* HO *scheme has very weak compromise resilience properties. An attacker that compromises any node will be able to forge tokens as if they came from an honest node that has access to ak, and these tokens appear valid to any other node. It becomes clear that this trade-off is inherent to these properties: the larger the group that a source* S *is indistinguishable from, the larger source nodes an attacker can compromise to forge messages from* S.

In our scheme illustrated in Figure 3.7, we circumnavigate this trade-off by exploiting U*'s role in authenticating* S*, at the expense of an additional message (c_1) to user* U*. In our construction, we leverage the role of* U*, as an intermediary that communicates with both* S *and* T *nodes, to add an additional layer of security that preserves both path privacy and group-compromise resilience. Our* StrongHO *scheme requires node* S *to generate signature σ_0 on user-communicated parameters ($epk_{\mathtt{U}} \| ctxt_{\mathtt{T}}$) as well as the authentication tag τ_1, which is subsequently encrypted to produce the ciphertext c_1. The encryption of σ_0 preserves the identity of* S *against any potential attacks to source privacy. Additionally, by verifying σ_0 and stripping it from the message,* S *is authenticated in a manner that secures both path privacy and compromise resilience. Moreover, the use of* PKE *to generate $ctxt_{\mathtt{T}}$ allows* U *to authenticate* T *while guaranteeing target privacy.*

3.4 Security Analysis

In this section we provide an analysis of the secure HO protocol that we introduced in Section 3.3. In particular, we provide a full proof for key indistinguishability of the StrongHO protocol to demonstrate how the analysis of a security property occurs in our framework. Next we provide proof sketches of all other properties of our StrongHO protocol.

We begin with proving the KIND security of the StrongHO protocol, described in Figure 3.7. The cryptographic assumptions that we use can be found in 2.2.

Theorem 1 (StrongHO KIND Security)**.** *The* StrongHO *protocol presented in Figure 3.7 is* KIND*-secure under cleanness predicate* $\mathsf{clean}_{\mathsf{str\text{-}kind}}$ *(capturing perfect forward security). That is, for any PPT algorithm* $\mathcal{A}$ *against the* KIND *security experiment (defined in Figure 3.3)* $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* pprf*,* sufcma*,* ind-1cca*,* ind-cca*,* ind-cca *and* sufcma *security of the* PRF*,* PPRF*,* MAC*,* KEM*,* PKE*,* AE *and* SIG *primitives respectively. Thus we have:* $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathcal{A},\mathsf{prf}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{pprf}}_{\mathsf{PPRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{MAC},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}1cca}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{PKE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{AE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{SIG},n_P,n_S}(\lambda)$.

Proof. We split the analysis into three cases:

- **Case 1:** Test session does not UT-match another session and $\pi^s_i.\rho = \mathtt{U}$
- **Case 2:** Test session does not UT-match another session and $\pi^s_i.\rho = \mathtt{T}$
- **Case 3:** Test session has a UT-matching session.

We proceed via a sequence of games. We bound the difference in the adversary's advantage in each game with the underlying cryptographic assumptions until the adversary reaches a game where the advantage of that game equals 0, which shows that adversary $\mathcal{A}$ cannot win with non-negligible advantage.

We begin by dividing the proof into three separate cases (and denote with $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_i}(\lambda)$ the advantage of the adversary in winning the key indistinguishability game in Case i) where the query $\mathsf{Test}(i,s)$ has been issued. It follows that $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_1}(\lambda) + \mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_2}(\lambda) + \mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_3}(\lambda)$

As shorthand we define with $\mathsf{Adv}^{\mathcal{A}}_{G_i}(\lambda)$ the advantage of $\mathcal{A}$ in Game i. We begin with Case 1.

Case 1: Test session does not UT-match another session and $\pi^s_i.\rho = \mathtt{U}$. By the definition of the case (and the cleanness predicate defined in Definition 26), we assume that the adversary $\mathcal{A}$ has not been able to compromise the bootstrap key bk of the Test session before the Test session completes, nor the long-term public key pk of the target session, nor the long-term public key of the source session. By the end of this case we show that there exists an honest session π^t_j such that $ctxt_{\mathtt{U}} \in \pi^s_i.T_{\mathtt{C}}$, $ctxt'_{\mathtt{U}} \in \pi^t_j.T_{\mathtt{C}}$, $ctxt_{\mathtt{U}} = ctxt'_{\mathtt{U}}$, and $epk_{\mathtt{U}} \in \pi^s_i.T_{\mathtt{C}}$, $epk'_{\mathtt{U}} \in \pi^t_j.T_{\mathtt{C}}$, $epk_{\mathtt{U}} = epk'_{\mathtt{U}}$.

Game 0 This is the initial KIND security game. Thus $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$

Game 1 In this game, we guess the index (i, s) of the session π_i^s, and abort if during the execution of the experiment, a query $\mathsf{Test}(i^*, s^*)$ is received and $(i^*, s^*) \neq (i, s)$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq n_P n_S \cdot \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda)$.

Game 2 In this game, we guess the index (j, t) of the source partner π_j^t, and abort if during the execution of the experiment, π_i^s $\mathtt{US}$-matches with some session $\pi_{j^*}^{t^*}$, but $(j^*, t^*) \neq (j, t)$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq n_S n_P \cdot \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda)$.

Game 3 In this game we introduce an abort event $\mathtt{event_S}$ that occurs if the Test session π_i^s sets $\alpha = \mathtt{accept}$ without an honest $\mathtt{US}$-match. In the following games, we bound this advantage and thus: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \Pr(\mathtt{event_S}) + \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda)$.

Game 4 In this game the challenger replaces the derived keys $mk, tk, id_{bk} = \mathsf{KDF}(bk, \epsilon)$ with uniformly random values $\widetilde{mk}, \widetilde{tk}, \widetilde{id_{bk}} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) by defining a reduction $\mathcal{B}_1$ that interacts with a PRF challenger. By definition of Case 1 and the cleanness conditions 4, 5 in Definition 26, $\mathcal{A}$ cannot issue $\mathsf{Compromise}$ queries before $\pi_i^s.\alpha = \mathtt{accept}$, since it accepts without a matching $\mathtt{UT}$ partner. Since by definition of our framework bk is already uniformly random and independent, this change is sound and introduces additional advantage bound by the PRF security. Specifically, if the test bit sampled by PRF challenger is 0, then $\widetilde{mk}, \widetilde{tk}, \widetilde{id_{bk}} = \mathsf{PRF}(bk, \epsilon)$ and we are in **Game 3**. If the test bit sampled by PRF challenger is 1, then $\widetilde{mk}, \widetilde{tk}, \widetilde{id_{bk}} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 4**. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda) + \mathsf{Adv}^{\mathcal{B}_1,\mathsf{prf}}_{\mathsf{PRF}}(\lambda)$.

Game 5 In this game, $\mathcal{C}$ aborts if the adversary $\mathcal{A}$ is able to produce a value $c_1 = \mathsf{AE.Enc}(\widetilde{tk}, \sigma_0 \| \tau_1)$ that decrypts correctly using $\widetilde{tk}$. Specifically, we introduce a reduction $\mathcal{B}_2$ that initializes an auth challenger $\mathcal{C}_{\mathsf{auth}}$. Whenever $\mathcal{C}$ is required to encrypt/decrypt using $\widetilde{tk}$, $\mathcal{B}_2$ instead queries $\mathcal{C}_{\mathsf{auth}}$'s respective encryption/decryption oracles. By **Game 4** we know that $\widetilde{tk}$ is a uniformly random and independent value, and thus this substitution of keys is undetectable. If $\mathcal{A}$ can provide a ciphertext c_1' that decrypts correctly, but was never output by $\mathcal{C}_{\mathsf{auth}}$, then it follows that $\mathcal{A}$ has forged a ciphertext c_1' breaks the auth security of the AE scheme as in Definition 9. We note that the ciphertext c_1 contains (and thus authenticates) the signature σ_0, which itself is computed over all messages received by $\mathtt{S}$ from the $\mathtt{U}$. Thus, $\mathtt{U}$ now aborts if they complete the preparation phase without a $\mathtt{US}$-matching partner, and $\Pr(\mathtt{event}_\alpha) = 0$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) + \mathsf{Adv}^{\mathcal{B}_2,\mathsf{auth}}_{\mathsf{AE}}(\lambda)$.

Game 6 In this game, $\mathcal{C}$ guesses the party index ℓ of the intended target partner of the test session π_i^s, and aborts if $\pi_i^s.pid \neq \ell$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) \leq n_P \cdot \mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda)$.

Game 7 In this game $\mathcal{C}$ introduces an abort event $\texttt{event}_\texttt{T}$ that occurs if the Test session π_i^s sets $\alpha = \texttt{accept}$ without an honest $\texttt{UT}$-match. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) \leq \Pr(\texttt{event}_\texttt{T}) + \mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda)$. We note that by definition of the case, π_i^s never has a $\texttt{UT}$-match and thus $\mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda) = 0$, since $\mathcal{A}$ can never test a session that aborts before $\alpha \leftarrow \texttt{accept}$. In what follows, we bound $\Pr(\texttt{event}_\texttt{T})$.

Game 8 In this game, $\mathcal{C}$ replaces ck in $ctxt_\texttt{T}$ computed by π_i^s with a random string of the same length $\widetilde{ck}$. We construct a reduction $\mathcal{B}_3$ that interacts with an $\mathsf{ind\text{-}cca}$ PKE challenger. At the beginning of the experiment, when $\mathcal{B}_3$ receives the list of public-keys $(pk_1, \ldots, pk_{n_P})$ from $\mathcal{C}$, $\mathcal{B}_3$ initializes a $\mathsf{ind\text{-}cca}$ challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces pk_l with pk output by $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. When π_i^s computes $ctxt_\texttt{T}$, $\mathcal{B}_3$ instead picks a uniformly random binary string z' of length equal to $z = id_{bk} \| ck$ and submits (z, z') to the $\mathsf{PKE.Enc}$ oracle. For any decryption operations requiring sk_ℓ, $\mathcal{B}_3$ submits the query to its respective Dec oracle, except for decrypting $ctxt_\texttt{T}$, where it simply sets the output to z. When the random bit b sampled by the PKE challenger is 0, $ctxt_\texttt{T}$ contains the encryption of z, so we are in **Game 7**, otherwise we are in **Game 8**. By definition of cleanness condition 26 of $\mathsf{clean}_{\mathsf{str\text{-}kind}}$ and Case 1, $\mathcal{A}$ cannot issue $\mathsf{Corrupt}(\ell)$ queries before $\pi_i^s.\alpha = \texttt{accept}$. Thus, $\mathcal{A}$ cannot know any information about ck, since it is never communicated to $\mathcal{A}$. Thus $\Pr(\texttt{event}_\texttt{T}) \leq \mathsf{Adv}^{\mathcal{A}}_{G_8}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{PKE},\mathcal{B}_3}()$.

Game 9 Similar to **Game 4**, in this game $\mathcal{C}$ replaces the derived keys $k, hk, mk' = \mathsf{KDF}(k_\texttt{U}, \widetilde{ck})$ with uniformly random values $\widetilde{k}, \widetilde{hk}, \widetilde{mk'} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ by defining a reduction $\mathcal{B}_4$ that interacts with a PRF challenger. By **Game 8** $\widetilde{ck}$ is already uniformly random and independent, thus this change is sound and introduces additional advantage bound by the PRF security. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_8}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda) + \mathsf{Adv}^{\mathcal{B}_4,\mathsf{prf}}_{\mathsf{PRF}}(\lambda)$.

Game 10 In this game, $\mathcal{C}$ introduces an abort event that triggers if $\mathcal{A}$ is able to successfully forge τ_2 to the Test session, without some honest $\texttt{T}$ session π that outputs τ_2. Specifically, $\mathcal{C}$ introduces $\mathcal{B}_5$ that initializes a MAC challenger $\mathcal{C}_{\mathsf{sufcma}}$. Whenever $\mathcal{B}_5$ is required to generate a MAC tag using $\widetilde{mk'}$, $\mathcal{B}_5$ instead queries $\mathcal{C}_{\mathsf{sufcma}}$, which is sound by **Game 9**. If π_i^s accepts a MAC tag τ_2 that was not produced by an honest target session, then $\mathcal{A}$ has broken the sufcma security of the MAC. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda) + \mathsf{Adv}^{\mathcal{B}_5,\mathsf{sufcma}}_{\mathsf{MAC}}(\lambda)$. Note that the MAC tag authenticates all messages sent in the Contact phase, so by **Game 10** π_i^s now aborts before accepting without a $\texttt{UT}$-match and thus $\mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda) = 0$.

We now transition to **Case 2**.

Case 2:Test session does not UT-match another session and $\pi_i^s.\rho = \mathtt{T}$. By the definition of the case, we assume that the adversary $\mathcal{A}$ has not been able to compromise the user partner's bootstrap key bk before the Test session completes, nor the long-term secret keys sk ak of the source session.

Game 0 This is the standard KIND security game. Thus $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$

Game 1 In this game, we guess the index (i, s) of the target session π_i^s, and abort if during the execution of the experiment, a query $\mathsf{Test}(i^*, s^*)$ is received and $(i^*, s^*) \neq (i, s)$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq n_S n_P \cdot \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda)$.

Game 2 In this game we replace the computation of ak_1 by π_i^s with uniformly random value $\widetilde{ak_1}$. Specifically, we define a reduction $\mathcal{B}_6$ that works as follows: At the beginning of the game $\mathcal{B}_6$ initializes a PPRF challenger $\mathcal{C}_{\mathsf{random}}$. Additionally, $\mathcal{B}_6$ maintains a lookup table PARTIES. Whenever, $\mathcal{B}_6$ needs to evaluate an input x on the puncturable state shared by all parties, $\mathcal{B}_6$ queries the lookup table on x. If an entry (P, out) returns, $\mathcal{B}_6$ checks if the current party calling PPRF.Eval is $i \in \mathsf{P}$. If so $\mathcal{B}_6$ aborts since it indicates that x has already been evaluated on the puncturable state. Otherwise, $\mathcal{B}_6$ uses out as the output value. If there exists no such entry, $\mathcal{B}_6$ queries $\mathsf{PPRF.Eval}(x)$ to $\mathcal{C}_{\mathsf{random}}$, replaces the computation of ak with the output value, and adds (i, out) to the PARTIES lookup table (where i is the party index). Whenever $\mathcal{B}_6$ needs to puncture on an input x, $\mathcal{B}_6$ queries the lookup table in x, recovering entry (P, out). If the current party i^* calling PPRF.Punc is $i^* \in \mathsf{P}$, then $\mathcal{B}$ aborts. Otherwise, $\mathsf{P} \xleftarrow{\mathsf{u}} i^*$ and $\mathcal{B}$ adds (P, out) under x. Finally, $\mathcal{B}$ replaces the computation of ak_1 in the Test session (and any session that computes ak_1) by calling the challenger's PPRF oracle $\mathsf{PPRF}.\mathcal{C}(epk_{\mathtt{U}} \| ctxt_{\mathtt{T}})$, returning a uniformly random $\widetilde{ak_1}$. If the bit b sampled by $\mathcal{C}_{\mathsf{random}}$ is 0, then we are in **Game 1**, otherwise we are in **Game 2**. We note that this is exactly how all parties engage with their collective PPRF state, and as such this replacement is sound. If $\mathcal{A}$ can distinguish between the two games, then $\mathcal{A}$ breaks the random game by Definition 6. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + \mathsf{Adv}^{\mathcal{B}_6,\mathsf{random}}_{\mathsf{PPRF}}(\lambda)$.

Game 3 This game proceeds identically to **Game 10** of **Case 1**. In this game $\mathcal{C}$ introduces an abort event that triggers if $\mathcal{A}$ is able to successfully forge τ_1 to the Test session, without some honest S session that outputs τ_1, by defining a reduction $\mathcal{B}_7$. Since $\widetilde{ak_1}$ is uniformly random and independent by **Game 2**. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) + \mathsf{Adv}^{\mathcal{B}_7,\mathsf{sufcma}}_{\mathsf{MAC}}(\lambda)$.

Game 4 In this game we guess the honest source session π_u^k that produced the MAC tag τ_1 which must exist by **Game 3**. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) \leq n_P n_S \cdot \mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda)$.

Game 5 In this game $\mathcal{C}$ replaces the derived keys $mk, tk, id_{bk} = \mathsf{KDF}(bk, \epsilon)$ with uniformly random values $\widetilde{mk}, \widetilde{tk}, \widetilde{id_{bk}} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ in π_u^k (and its corresponding user session) by defining a reduction $\mathcal{B}_8$ that interacts with a PRF challenger as in **Case 1 Game 4**. By definition of Case 2 and the cleanness condition 6 in Definition 26, $\mathcal{A}$ cannot issue relevant Compromise queries. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) + \mathsf{Adv}^{\mathcal{B}_8,\mathsf{prf}}_{\mathsf{PRF}}(\lambda)$.

Game 6 In this game, $\mathcal{C}$ introduces an abort event that triggers if $\mathcal{A}$ is able to successfully forge τ_0 to the guessed source session π_u^k, without some honest U session that outputs τ_0. $\mathcal{C}$ does so by introducing a reduction $\mathcal{B}_{10}$ that interacts with a MAC challenger $\mathcal{C}_{\mathsf{sufcma}}$ as in **Case 1**, **Game 10**. Thus $\mathcal{A}$ cannot modify messages to π_u^k and we have $\mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) + \mathsf{Adv}^{\mathcal{B}_{10},\mathsf{sufcma}}_{\mathsf{MAC}}(\lambda)$. By **Game 6** we know that there exists an honest user session that communicated with π_u^k without modification. This π_u^k produced τ_1 honestly, which authenticates the public key $epk_{\mathtt{U}}$ and ciphertext $ctxt_{\mathtt{T}}$ received by the target session π_i^s. Thus, by **Game 6** we have that there exists some honest user session that UT-matches π_i^s, and by definition of Case 2 $\mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) = 0$.

Now we transition to **Case 3.**

Case 3: Test session has a UT-matching session.

Game 0 This is the initial KIND security game. Thus $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_3}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 In this game, $\mathcal{C}$ guesses the index of the test session (i, s) and the is UT matching partner π_j^t and aborts if their guess was incorrect. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq {n_P}^2 n_S^2 \cdot \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda)$.

Game 2 In this game, $\mathcal{C}$ replaces the key $k_{\mathtt{U}}$ derived in the test session π_i^s with the uniformly random and independent value $\widetilde{k_{\mathtt{U}}}$. WLOG we assume that $\pi_i^s.\rho = \mathtt{U}$, but the same argument (modulo switching between π_i^s and π_j^t) applies if $\pi_i^s.\rho = \mathtt{T}$. $\mathcal{C}$ defines a reduction $\mathcal{B}_{11}$ that interacts with an ind-1cca KEM challenger, replacing the $epk_{\mathtt{U}}$ generation by π_i^s (resp. π_j^t), and the ciphertext $ctxt_{\mathtt{U}}$ sent by π_j^t (resp. π_i^s) with $\widetilde{epk_{\mathtt{U}}}$ and ciphertext $\widetilde{ctxt_{\mathtt{U}}}$ received from the ind-1cca KEM challenger, and the computation of $k_{\mathtt{U}}$ with the output key. In the event the adversary sends an incorrect $ctxt'_{\mathtt{U}}$, the user U will query challenger $\mathcal{C}$'s Decaps oracle which will return some key k. Detecting the replacement of $k_{\mathtt{U}}$ implies an efficient distinguishing PPT algorithm $\mathcal{A}$ against ind-1cca security of KEM. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + \mathsf{Adv}^{\mathcal{B}_{11},\mathsf{ind\text{-}1cca}}_{\mathsf{KEM}}(\lambda)$.

Game 3 In this game, $\mathcal{C}$ replaces the derived keys $k, hk, mk', tk' \xleftarrow{\$} \mathsf{KDF}(\widetilde{k_{\mathtt{U}}}, ck)$ with uniformly random values $\widetilde{k}$, $\widetilde{hk}$, $\widetilde{mk'}$ $\widetilde{tk'} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ by defining a reduction $\mathcal{B}_{12}$ that interacts

with a PRF challenger as in **Game 5** of **Case 2**. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda)+\mathsf{Adv}^{\mathcal{B}_{12},\mathsf{prf}}_{\mathsf{PRF}}(\lambda)$,.

Here we emphasize that as a result of these changes, the session key $\widetilde{k}$ and the handover key $\widetilde{hk}$ are now both uniformly random and independent of the protocol execution regardless of the bit b sampled by $\mathcal{C}$, thus $\mathcal{A}$ has no advantage in guessing the bit b. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) = 0$. □

Next, we turn our attention to the UNLINK, SPRIV and TPRIV notions of our StrongHO construction and provide brief proof sketches for each notion.

Theorem 2 (StrongHO UNLINK Security)**.** *The* StrongHO *protocol presented in Figure 3.7 is* UNLINK*-secure under cleanness predicate* $\mathsf{clean}_{\mathsf{str\text{-}unlink}}$*. That is, for any PPT algorithm* $\mathcal{A}$ *against the* UNLINK *security experiment (defined in Figure 3.4),* $\mathsf{Adv}^{\mathsf{UNLINK},\mathsf{clean}_{\mathsf{str\text{-}unlink}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* pprf*,* sufcma*,* ind-1cca*,* ind-cca*,* ind-cca *and* sufcma *security of the* PRF*,* PPRF*,* MAC*,* KEM*,* PKE*,* AE *and* SIG *primitives respectively.*

Proof. Here we provide a proof sketch. In StrongHO there are only two values that are linked to other sessions owned by the same test session π^s_i - the bootstrap key bk, which may be shared with some previous handover user session owned by party i, and the handover key hk, which might be re-used in some future user session owned by party i. All other values are generated independently of all other sessions by the user. Thus, we must prove that bk in the previous user session, and hk in the test session π^s_i are completely independent from other sessions owned by the same user.

We can use the proof of KIND security to replace hk with a uniformly random and independent value $\widetilde{hk}$ in the previous session. This is sufficient to show that the bootstrap key bk used in the test session π^s_i is independent of hk computed in the previous session. Similarly, we can use the proof of KIND security to replace the computation of hk in the test session, which is sufficient to show that the bootstrap key used in some proceeding session is independent of the handover key used in the test session - again, this corresponds exactly with proving KIND for the test session and its previous session (if any exist). Thus, incurring a factor of 2, by the same arguments as in the KIND proof of StrongHO, StrongHO achieves UNLINK security. Thus we have: $\mathsf{Adv}^{\mathsf{UNLINK},\mathsf{clean}_{\mathsf{str\text{-}unlink}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda) \leq 2 \cdot \Big(\mathsf{Adv}^{\mathcal{A},\mathsf{prf}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{pprf}}_{\mathsf{PPRF},n_P,n_S}(\lambda)+\mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{MAC},n_P,n_S}(\lambda)+\mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}1cca}}_{\mathsf{KEM},n_P,n_S}(\lambda)+\mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{PKE},n_P,n_S}(\lambda)+\mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{AE},n_P,n_S}(\lambda)+ \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{SIG},n_P,n_S}(\lambda) \Big)$.

□

Theorem 3 (StrongHO SPRIV Security)**.** *The* StrongHO *protocol presented in Figure 3.7 is* SPRIV*-secure under cleanness predicate* $\mathsf{clean}_{\mathsf{str\text{-}spriv}}$*. That is, for any PPT algorithm* $\mathcal{A}$ *against the* SPRIV *security experiment (defined in Figure 3.6)* $\mathsf{Adv}^{\mathsf{SPRIV},\mathsf{clean}_{\mathsf{str\text{-}spriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* pprf*,* sufcma*,* ind-1cca*,* ind-cca*,* ikcca *and* sufcma *security of the* PRF*,* PPRF*,* MAC*,* KEM*,* PKE*,* PKE *and* SIG *primitives respectively.*

Proof. Here we provide a proof sketch. We note that the only value output by the source is c_1, which contains σ_0, τ_1. Since τ_1 is computed from ak, (which all source parties share), this cannot be used to distinguish the source $\mathtt{S}$. The signature σ_0, however is signed using the public long-term signing key $spk_{\mathtt{S}}$ of $\mathtt{S}$, which could reveal the identity of the $\mathtt{S}$ to a potential adversary. However, we encrypt σ_0 along with τ_1, and thus only the $\mathtt{U}$ who shares tk with the source learns the identity of $\mathtt{S}$. Therefore, any modifications to c_1 will break the AE.auth security of our construction.

However, the user does use the long-term public key $epk_{\mathtt{S}}$ of the source to encrypt the bootstrap key identifier id_{bk}. Thus, we must argue that the ciphertext itself cannot leak information about the identity of the source. Since the bootstrap key identifier id_{bk} is computed from the bootstrap key bk, by proving that the initial bootstrap key bk used by the source is uniformly random and independent (either by definition of the framework, if the source was not bootstrapped from some previous target session, or via a KIND argument for the previous session where the source acted as a target), then we can iteratively replace id_{bk} with a uniformly random value $\widetilde{id_{bk}}$ and this does not link to a previous session. However, the ciphertext itself might identify the source. Thus, we replace the generation of ciphertexts $ctxt_{\mathtt{S}}$ by initializing a PKE ikcca challenger for the public keys of the test session π_b's owner party $\pi_b.spid$ and the other adversarially-nominated session $\pi_{b'}$'s owner party $\pi_{b'}.spid$. By the ikcca security of the PKE any adversary that is capable of associating the public key of $\mathtt{S}$ with $ctxt_{\mathtt{S}}$ can also be used to break the ikcca-security of the PKE scheme. Thus, by the same arguments as in the KIND proof of StrongHO and the ikcca security of the PKE scheme, $\mathcal{A}$ has negligible advantage in breaking SPRIV security. Thus we have: $\mathsf{Adv}^{\mathsf{SPRIV},\mathsf{clean}_{\mathsf{str\text{-}spriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathcal{A},\mathsf{AUTH}}_{\mathsf{AE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ikcca}}_{\mathsf{PKE},n_P,n_S}(\lambda) + 2 \cdot \Big(\mathsf{Adv}^{\mathcal{A},\mathsf{prf}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{pprf}}_{\mathsf{PPRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{MAC},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}1cca}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{PKE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{AE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{SIG},n_P,n_S}(\lambda) \Big)$. □

Theorem 4 (StrongHO TPRIV Security)**.** *The* StrongHO *protocol presented in Figure 3.7 is* TPRIV*-secure under cleanness predicate* $\mathsf{clean}_{\mathsf{str\text{-}tpriv}}$*. That is, for any PPT algorithm* $\mathcal{A}$ *against the* TPRIV *security experiment (defined in Figure 3.5)* $\mathsf{Adv}^{\mathsf{TPRIV},\mathsf{clean}_{\mathsf{str\text{-}tpriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* pprf*,* sufcma*,* ind-1cca*,* ind-cca*,* ikcca *and* sufcma *security of the* PRF*,* PPRF*,* MAC*,* KEM*,* PKE*,* PKE *and* SIG *primitives respectively.*

Proof. We provide here a proof sketch. We note that the only values that the target party uses across multiple sessions is the handover key hk derived in this session, and the long-term PKE public key. We note that we can replace the handover key hk in this protocol execution with a uniformly random value $\widetilde{hk}$ by the KIND security of the StrongHO protocol. Also, similarly to the SPRIV security analysis of the StrongHO protocol, we can argue that the ciphertext generated by the user (encrypting the fresh entropy ck) can be used to identify the target by the key indistinguishability of the PKE scheme. Thus, by the same arguments as in the KIND proof of StrongHO and the ikcca security of the PKE scheme, $\mathcal{A}$ has negligible advantage in breaking TPRIV security. Thus we have: $\mathsf{Adv}^{\mathsf{TPRIV},\mathsf{clean}_{\mathsf{str\text{-}tpriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathcal{A},\mathsf{ikcca}}_{\mathsf{PKE},n_P,n_S}(\lambda) + \Big(\mathsf{Adv}^{\mathcal{A},\mathsf{prf}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{pprf}}_{\mathsf{PPRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{MAC},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}1cca}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{PKE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{ind\text{-}cca}}_{\mathsf{AE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathcal{A},\mathsf{sufcma}}_{\mathsf{SIG},n_P,n_S}(\lambda)\Big)$. □

Thus, by Theorem 3 and Theorem 4, we can conclude that StrongHO achieves PPRIV. We formalise this in Theorem 5.

Theorem 5 (StrongHO PPRIV Security)**.** *The* StrongHO *protocol presented in Figure 3.7 is* PPRIV*-secure.*

Proof. The proof follows from the proofs of Theorem 3 and Theorem 4. Thus we have : $\mathsf{Adv}^{\mathsf{PPRIV},\mathsf{clean}_{\mathsf{str\text{-}ppriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda) = \max\{\mathsf{Adv}^{\mathsf{SPRIV},\mathsf{clean}_{\mathsf{str\text{-}spriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda), \mathsf{Adv}^{\mathsf{TPRIV},\mathsf{clean}_{\mathsf{str\text{-}tpriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)\}$. □

3.5 Conclusion

Secure handovers are critical for LDACS/CPDLC avionic communication to ensure uninterrupted and secure data exchange when aircrafts transition between ground stations, and they are equally integral to modern 5G networks, enabling seamless and secure connectivity as devices move across network nodes. In this chapter, we introduced the first formalization

of secure handovers, distinguishing handovers as a unique primitive rather than conflating them with related and often interchangeably discussed concepts such as key exchanges. The need to understand the unique characteristics of handovers to analyze and enhance existing secure handover schemes, as well as to design new constructions within the constraints of avionic communication, serve as the primary motivation for our formalization. We identify key security notions that are essential for a secure handover scheme and emphasize that our formalization was instrumental in their discovery. For example, incorporating our notion of *path privacy* would not be feasible within a key exchange model, as such models are not designed to support the secure transition of an ongoing session between parties. We designed StrongHO as a secure HO within our framework, exemplifying the highest level of security achievable within our model. StrongHO encapsulates all the security notions identified for a secure handover scheme, with its security validated through formal proofs.

Part III

Analysis of Deep Space Protocols

Chapter 4

Analysis of BPSec

The BPSec secure channel environment, security expectations, and functionality are quite unlike traditional secure channel protocols, creating a non-trivial environment for analysis. In this chapter, we formalize the complex goals of BPSec and provide an analysis of the protocol. This includes a model and proof corresponding to the type of security BPSec can be claimed to offer. We note issues in BPSec and outline normative security goals that it cannot assure. Furthermore, we offer recommendations to enhance security guarantees within the intended design constraints of BPSec by introducing StrongBPSec, which reinforces the integrity assurances of BPSec. StrongBPSec accomplishes this by maintaining a verifiable ledger of changes, allowing the final acceptor of a bundle to independently verify the integrity of modifications made throughout a bundle's journey. We also present a stronger model and analysis to match the improved security of StrongBPSec.

4.1 Introduction

Recent years have seen a significant resurgence of interest in space exploration, driving a fervent pursuit of advancements in deep space missions. This renewed interest in space exploration across commercial and scientific communities alike, exemplified by SpaceX's ambitious projects like Starship [118] and lunar missions [66], underscores a growing commitment to advancing both scientific discovery and commercial opportunities in space. Moreover, the Moonlight Lunar Communications and Navigation Services (LCNS) project was recently launched by the European Space Agency (ESA). This initiative aims to facilitate high-speed data transmission between Earth and the Moon to support an estimated

400 lunar missions over the next two decades [66]. However, data communication within deep space is uniquely challenging compared to their terrestrial counterpart. In Table 4.1 we summarize these challenges and draw your attention to the demanding restraints unique to deep space communication.

Terrestrial Internet	**Interplanetary Internet**
Continuous availability of links and devices	Discontinuous availability
Network topology changes are slow and predictable	Constant orbital movement of entities (e.g., satellites)
Low latency and fast RTT, e.g., RTT between two points is approximately 100 to 300 ms	Extreme distances, e.g., RTT between Earth and Mars is around 24 minutes on average
Low packet loss rate	High rates of data loss (radio signal interference, Van Allen Belt radiation)
Bi-directional data links	Data links are often simplex
Ubiquitous client-server model	Handshakes and negotiations are unreliable/costly

Table 4.1: Challenges to deep space Communication

To address the challenges outlined in Table 4.1 and meet the demanding constraints inherent in deep space communication, Delay Tolerant Networks (DTNs) were introduced as an effective solution. Briefly, DTN has been built to facilitate for extreme distances between communicating parties and any long and variable delays between data transmissions. In contrast to the end-to-end architecture of the terrestrial Internet, where the network passively forwards packets to their destination, nodes within Delay Tolerant Networks are capable of actively creating, modifying, sending, forwarding, and receiving messages. The functionality of active intermediate nodes, which partially process and forward messages towards their final destination, combined with the modular structure of DTN messages, is fundamental to the DTN architecture. Together, it addresses the challenges of simplex communication links, vast distances between nodes, and extreme delays by enabling nodes to include sufficient contextual information in sent or forwarded messages, ensuring that receiving nodes can process them correctly despite network disruptions.

4.1.1 Secure Deep Space Communication

A variety of Delay Tolerant Network (or DTN) protocols for space communications exist and are managed by the Consultative Committee for Space Data Systems (CCSDS) [3], including the Space Packet Protocol [2] and CCSDS File Delivery Protocol (CFDP) [1], but relatively little has been done in terms of cryptographic analysis of their security. The suite of DTN protocols are focused on a store-and-forward approach to make them more robust to environmental disruptions and message relay issues stated in Table 4.1. Since ground stations may also require substantive planning in the order of days to send messages [83], it is essential that each transmission has the capability of aggregating message information along its path and processing as needed. Space link efficiency is a particular concern for DTN protocols.

A survey of DTN key management protocols [89] reveals that formal cryptographic analysis in the provable security sense is relatively rare. Of the ones that have received analysis in the areas of identity-based cryptography, non-interactive key exchange, and group key management [10, 68, 109, 80, 5, 134], none are documented or promulgated by public institutions such as the IETF or CCSDS as deployed in practice, including in space communications. To elaborate, the identity-based cryptographic solutions proposed by Asokan et al. [10], Kate et al. [68] and Ahmad et al. [5] focus only on the initial key agreement and key management for DTN protocols and do not formally verify the security of their proposed constructions. The non-interactive key establishment construction proposed by Lv et al. [80] also limit the scope of their construction to the initial key establishment stage and again do not formally verify the security of their proposed protocol. Only the group key management construction proposed by Zhou et al. [134] formally verify the security of their construction. Finally, the work of Rüsch et al. [109] somewhat differs from the work so far discussed in that they focus on integrating forward secrecy guarantees for secret keys after an initial key establishment. They propose the use of puncturable encryption for this purpose but do not provide formal security proofs for their proposed construction.

Other DTN protocols that have known space uses, such as the Licklider Transmission Protocol, lack cryptographic security analysis [36]. Broadly speaking, the lack of cryptographic formal or computational analysis in this field leaves claims of security by various DTN protocols inconclusive, threat models underdefined, and security assumptions on the underlying cryptographic primitives unknown. This presents a gap that we address in this chapter, providing a first rigorous cryptographic analysis of DTN protocol BPSec.

4.1.2 Bundle Protocol Security

Originally developed for deep space communications, the Bundle Protocol Security (BPSec) [28] supports DTNs [27] as an application layer secure channel protocol to facilitate a store-and-forward paradigm for sending messages between nodes. It was specifically designed with space system security in mind, a use case where security protocols used on the Internet today, such as IPSec [56] and TLS [102], are sub-optimal as outlined in Table 4.1, due to latency, delays, and bandwidth constraints. BPSec was designed to be more suitable for distressed environments where delivery is not guaranteed, bandwidth is a premium, and roundtrip times are on the order of minutes or even hours. This has made it suitable to uses outside of deep space networks and even outside of space networks generally, with applications extending to the Internet of Things (IoT) [23].

In Figure 4.1 we present a simplified illustration of a BPSec use case in space. We particularly draw attention to the unique nature of the network infrastructure BPSec operates on, in comparison to the terrestrial Internet. In this example, the satellites orbiting earth act as bundle protocol agents (BPA) or nodes who create, forward and receive messages. The movement of these satellites and their availability to forward and receive messages are in a constant state of change which in turn creates a network infrastructure prone to high latency, connection disruptions and propagation delays. For instance in Figure 4.1, while the source node creates the message at t_0, due to the unavailability of a viable path, it cannot be forwarded to the intended destination node immediately and thus stores the message until t_1. Within such DTN infrastructures, participating nodes store messages for long periods and forward them once an appropriate link becomes available, until the message reaches its final destination.

4.1.3 BPSec Overview

In this section we describe core components of BPSec RFC 9172 [28] and the underlying Bundle Protocol version 7 RFC 9171 [35]. This is a high-level description; a detailed algorithmic description of BPSec is presented later in Figures 4.11, 4.12 and 4.13 which is based on the mandatory minimum Default Security Context [29].

Underlying BPSec is the Bundle Protocol (BP), a DTN protocol analogous to the networking layer Internet Protocol (IP) of the terrestrial Internet. NASA has included Bundle Protocol version 7 in its three-phased approach for rolling out DTN-based communication links by 2030 [84]. Similar to how IP delivers packets, BP create, forward, modify and

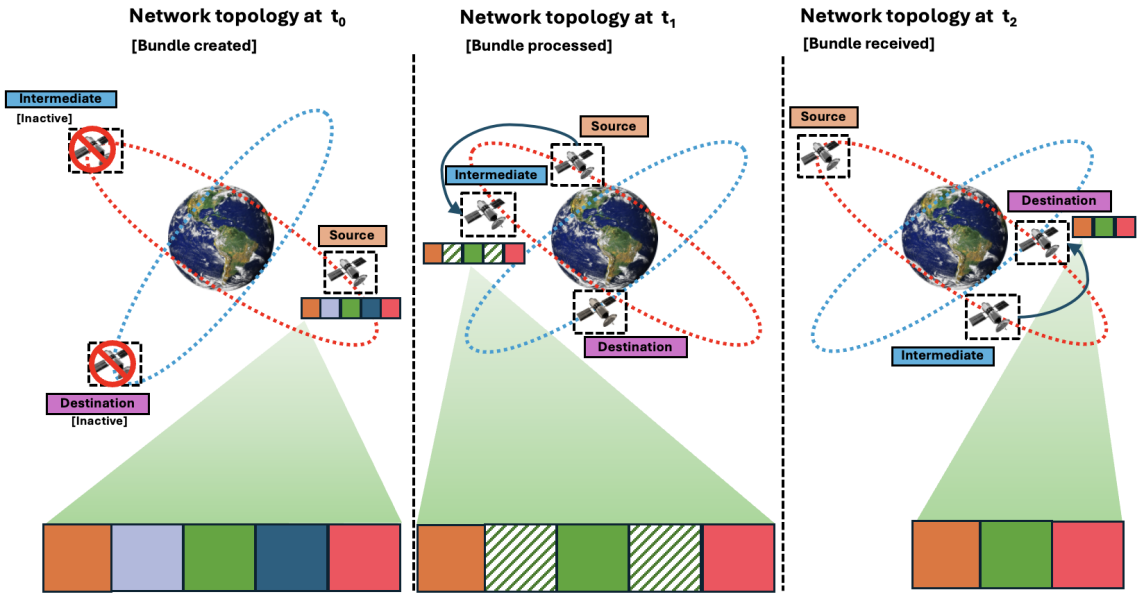


Figure 4.1: An execution of BPSec protocol. Source creates and forwards a bundle to intermediary. Intermediary processes blocks from the bundle, forwarding the modified bundle to destination. This processing and discarding of bundle blocks by an intermediate node is legitimate behavior within a BPSec execution, provided that the corresponding intermediary has been authorized to process bundles by the relevant policies.

receive *bundles*. Bundles are analogous to packets but differ in that they are designed to contain more information to facilitate their correct processing, in the absence of bi-directional links and retransmissions. A single bundle consists of many blocks; a mandatory primary block containing routing information; a mandatory payload block; and optional extension blocks containing additional data. Next, we identify the distinct roles of nodes participating in a BPSec channel and further elaborate on BPSec bundle block types.

Entities in a BPSec Channel The *Bundle Protocol Agent* (BPA) is described as a node component that offers the Bundle Protocol services and executes its procedures. We employ a slight abstraction of this for simplicity, and refer to the BPA as the node itself. Possible BPAs may include the *Source Node* (SN) that is the originator of the bundle and the *Destination Node* (DN) that receives the bundle which is the intended recipient. *Intermediate Nodes* (IN) may also receive and process bundles— potentially discarding and/or adding component blocks— before forward transmission. If a BPA $\in$ {SN, IN} adds a security block to a bundle (i.e., that it adds encryption or authentication) it is also called a *security source*. Similarly, if a BPA $\in$ {IN, DN} removes a security block (e.g.

decrypting a BCB target) it is also called a *security acceptor*. If the same BPA does not remove the block (e.g. verifying the MAC of a BIB), it is called a *security verifier*. We give the high-level bundle components below and provide a generic illustration in Figure 4.2:

- *Primary Block* The primary block carries information about the SN, DN, and lifespan of the bundle among other basic bundle identification and forwarding values. There is exactly one primary block per bundle, followed by possible extension blocks and finally a payload block.
- *Extension Block* An extension block is meant to provide additional functionality for bundle processing through annotative information (e.g. previous node block, bundle age block, abstract security block, etc.). Block Integrity Block (BIB) and Block Confidentiality Block (BCB) are extension blocks that have identical structures as abstract security blocks— both delineating ciphers, parameters, etc. for the target blocks they apply to. INs may also add extension blocks.
- *Payload Block*: There is one payload bock per bundle, always the last block in the bundle, which carries the application data. Since the Bundle Protocol can be used as an encapsulating protocol for another protocol (e.g., an application layer protocol and data) a payload block may be a *partial payload* or *fragmented payload* containing only a segment of the overall payload.
- *Target Block*: The block within a bundle to which a security operation is applied.
- *Security Context*: A security context includes [1]
  1. key origin (e.g. preshared/keywrapped keys)
  2. algorithms (e.g. AES-GCM, HMAC-SHA2)
  3. configurations
  4. relevant security policies

Analogous to how IPSec extends the IP protocol to integrate security guarantees in the networking layer, BPSec extends BP to incorporate confidentiality and integrity guarantees in the application layer. The BPSec protocol [28] assumes secure preshared keys/symmetric keys and consider key establishment and management as out of scope for its construction. BPSec adapts existing extension blocks into *security blocks* that provide additional security guarantees to bundles in transit.

[1] Security contexts are user-defined but RFC 9173 specifies mandatory-to-implement security contexts.

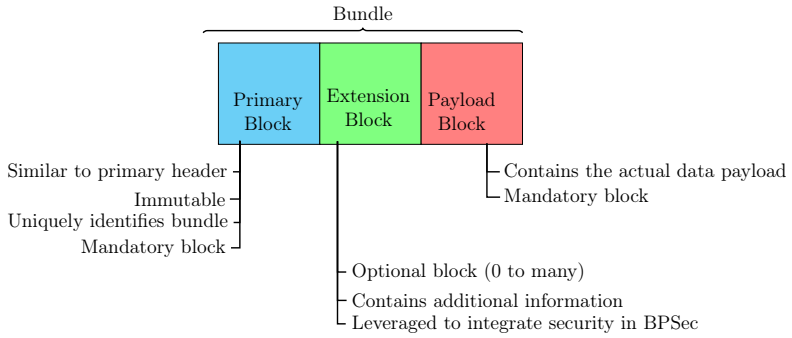


Figure 4.2: Generic bundle message format

BPSec introduces two types of security blocks; Bundle Confidentiality Block (BCB) for authenticated encryption; and Bundle Integrity Block (BIB) for data authenticity. The security blocks protect other *target* blocks and carry either a result of a cryptographic operation (e.g. MAC tag τ) or a pointer to a *target* block it has encrypted along with the necessary ciphersuite information. The Default Security Context for BPSec [29] instantiates BPSec under pre-shared key and key-wrapping assumptions, with HMAC-SHA2 for BIB security and AES-GCM for BCB security. Figure 4.3 illustrates the format of a BPSec bundle.

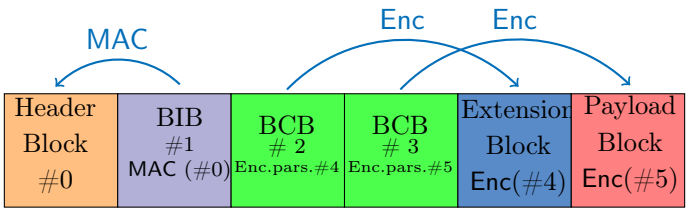


Figure 4.3: BPSec bundle message format

As shown in Table 4.1, domain-specific constraints in Delay Tolerant Networks (DTNs) hinder end-to-end key negotiations and handshakes. Consequently, BPSec does not aim to provide key agreement or established freshness of a session in entity authentication. Instead, BPSec aims to offer a *secure channel protocol* under the assumption of pre-established keys. The BPSec secure channel differs fundamentally from a secure channel on the terrestrial Internet, as it permits authorized intermediaries to modify bundle messages during their

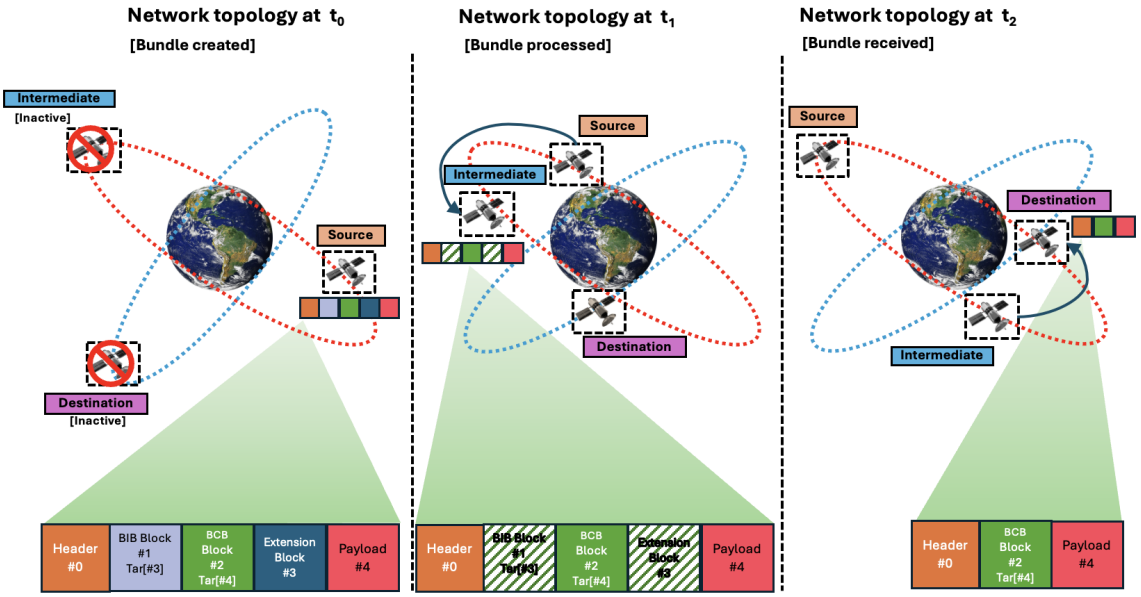


Figure 4.4: An expected BPSec execution in space.

transit between the originating source and the final destination. In the absence of the common handshake component, if BPSec only offers data authentication and/or confidentiality, one could ask why the protocol exists, e.g., instead of just applying ciphersuites to Bundle Protocol messages as needed. This question points towards the subtlety of BPSec; namely that it aims to achieve these properties across potentially multiple participants, even when intermediate nodes can add more data to the transmission or discard prior data, which may have been corrupted. The reader can here think of information distribution among satellites, where intermediate satellites must relay data or even add to a message, while it is also possible for data to be corrupted in transit, necessitating partial message discardment so as to preserve bandwidth. BPSec thus aims to define a channel security protocol that can modularly add and remove data, even while achieving its security goals. Figure 4.4 illustrates an expected generic execution of BPSec in space.

Important BPSec Design Decisions through Example BPSec focuses on *block-level granularity and interactions*. Security operations are applied to individual blocks within a bundle according to the security context of a BPA. In this way, intermediate nodes INs between a source node SN and destination node DN can add security operations just like a security source, or indeed remove security as if they were an acceptor for the bundle.

Figure 4.5 shows an IN, Charlie, adding BCBs to the bundle it received, acting as a relay from Alice to Bob. While Alice only provided authentication to Block 2 and 4 (creating a BIB block 1 that contains the applicable MAC tags and algorithm information), Charlie decides to AEAD-encrypt certain blocks as well, namely *target blocks* 1, 2, and 4. These are Alice's original blocks including the BIB itself. We also demonstrate a separate BCB in Block 6 that targets only Block 3. It is not consolidated with BCB Block 5 because it uses a different parameter set. Moreover, this AEAD encryption may have been realized with different instantiations (e.g. encrypt-then-MAC mode vs AESGCM); in such cases, BPSec requires the MAC tag output from the BCB to be placed in the BCB block, as opposed to being part of the ciphertext in Block 3.

If Charlie shares the same security context and secret keys with Alice and Bob, then Bob will be able to decrypt and verify the bundle upon receipt (if the bundle is delivered honestly). Otherwise, if the DN Bob does not have the requisite keys, at least one other IN will be needed to process the BCB blocks for Bob, i.e., to act as a *security acceptor*. Bundle encapsulation, whereby a bundle is encapsulated as a payload of a wrapping bundle, is recommended for cases where the bundle may arrive at the DN before being processed by a security acceptor.

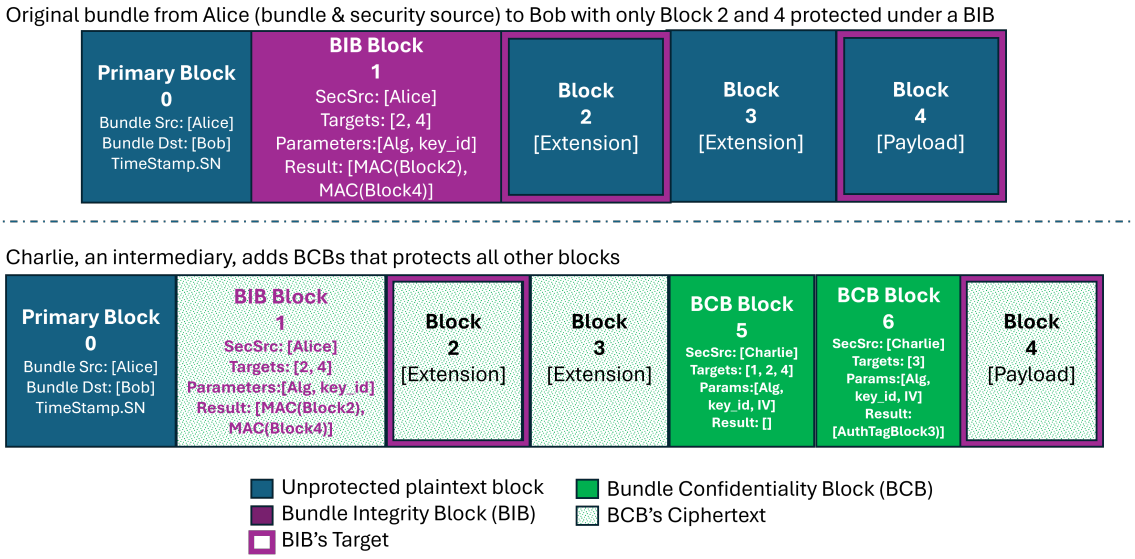


Figure 4.5: BPSec example showing block interactions between BCB and BIB made by different BPA security sources, Alice and Charlie, for a bundle intended for Bob.

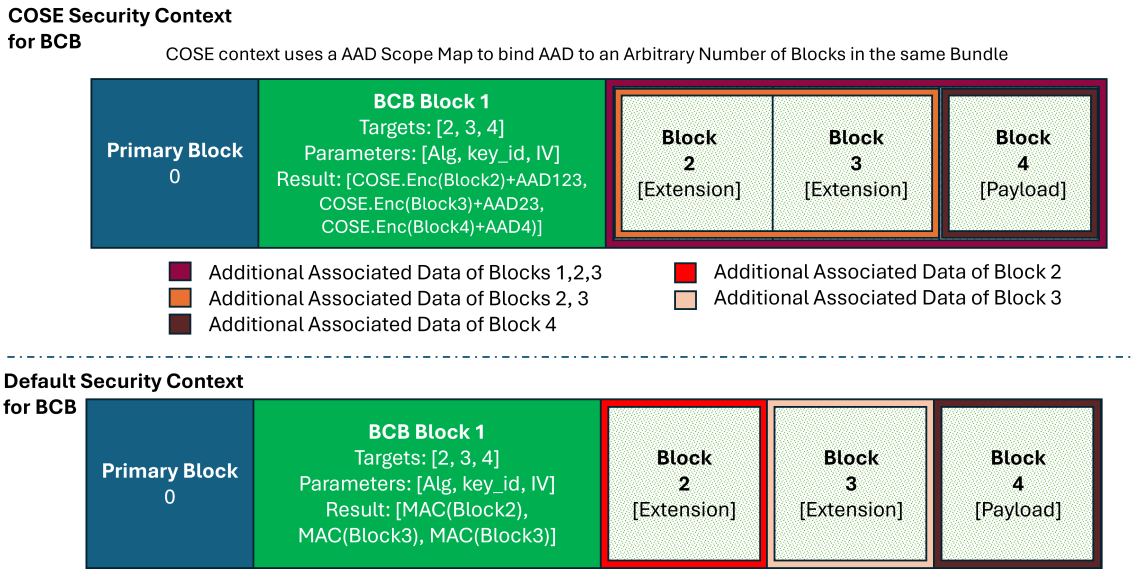


Figure 4.6: COSE Context vs Default Context treatment of AAD

4.1.4 BPSec COSE Context

BPSec's Default Security Context described in [29] provides the minimum level of security based on preshared symmetric keys to use within the bundle protocol. In an effort to introduce interoperability with other networks using DTN protocols and compatibility with asymmetric-keyed algorithms, the CBOR Object Signing and Encryption (COSE) Security context was proposed in the DTN IETF working group [114]. Some highlights of this draft standard include: defining how to incorporate signing and encryption to BPSec using PKI; expanding the additional associated data (AAD) to support binding AAD to an arbitrary number of blocks in the same bundle; and use of PKIX certificate for entity authentication.

We emphasize that while incorporating a Public Key Infrastructure (PKI) would significantly enhance the security of BPSec, the inherent challenges of key negotiation, management, and communication with trusted authorities will prove difficult within the constraints of deep space networks. Furthermore, we argue that binding multiple blocks with associate data may hinder the ability of legitimate intermediate nodes to partially process blocks bound in such manner.

4.1.5 Key Wrapping in BPSec

BPSec [29] offers the option to incorporate AES Key Wrapping (AESKW) in security blocks, according to the AESKW standards outlined in [63]. In an early request for proposal [51] for ANS X9.102 standard that discuss *key-wrapping* as a primitive, the goal of key wrapping was highlighted as "to protect the confidentiality and integrity of cryptographic keys without the use of nonces". Accordingly, BPSec's use of AESKW aims to enable secure sharing of cryptographic keys used within BIB/BCBs with other nodes who have access to the correct key wrapping keys. This approach helps mitigate the need for key negotiation, which is challenging in DTN environments due to the constraints of simplex communication links and the vast distances between nodes.

The AESKW wrapping algorithm takes as input a pre-established long-term key encryption key (KEK) k_{KEK}, a message to encrypt which in this case is the ephemeral key k, and optional associated data concatenated with a static integrity check vector (an ICV) which are passed into a six-round non-standard Feistel-network [108]. This outputs a ciphertext which is sent along to a receiver with the plaintext authenticated data. The unwrapping algorithm takes as input the k_{KEK}, ciphertext, ICV and authenticated data and outputs either the shared key k or error upon integrity check failure.

A serious caveat with AESWK and the three other *key-wrapping* schemes discussed in ANS X9.102 is that their security has not been formally proven. Rogaway and Shrimpton in [108] likens the primitive to a secure enciphering scheme similar to a strong, variable-input-length pseudo random permutation. However, they stop short of formally verifying the security of AESWK due to many ambiguities and lack of specificity regarding its construction in ANS X9.102. Instead, they introduce a novel framework called *deterministic authenticated encryption* (DAE) [108] that is capable of capturing the security goals of *key-wrapping*, which we leverage in the security analysis of BPSec and our StrongBPSec construction, a stronger variant of BPSec, in Section 4.6.

4.1.6 Key Management

Key management (key derivation, key exchange, key revocation, key separation) policies are not explicitly defined by any of the three relevant RFCs for BPSec. Instead, it is assumed that these are handled separately as part of network management [28]. RFC9173 [29] stipulates that BP nodes using security contexts need to "establish shared key encryption keys (KEKs) with other nodes in the network using an out-of-band mechanism". BPSec

does not provide key establishment or entity authentication mechanisms internally.

Symmetric keys were chosen over asymmetric keys (e.g. BIB with HMAC-SHA2) "in order to create a security context that can be used in all networking environments" [27]. RFC9173 stipulates that different keys must be used to perform different security operations (e.g. a separate keys for data encryption and integrity protection) and across different cipher-suites for the same operations (i.e. using separate keys for AES-GCM and AES-CBC). Depending on how symmetric keys are distributed for a given security context (key exchange, pre-shared, out-of-band), the use of ephemeral keys through AES-Key wrapping or a key-rotation policy must be used to curtail key leakage. Initialization vectors must also not be reused with the same key across multiple encryption operations.

4.1.7 BPSec: A Flexible Secure Channel

Typically, a secure channel is formed between two communication parties who have previously established a shared key. The security provided by a secure channel construction predominantly focuses on the *confidentiality* and *integrity* of transmitted data, i.e. only the sender and their respective receiver should be able to read and/or modify the data transmitted. To this extent, there is a rich body of work that formally captures the notions of secure channels as standalone primitives. The seminal works of Bellare et al. [21] and Rogaway [105] introduced the notions of *stateful authenticated encryption* and *authenticated encryption with associated data*, respectively, formalizing early ideas on secure channels. Protections against message replays, reordering, and dropping were early features in [21]. More recently, work has emphasized and differentiated the nature of data transmitted within secure channels, between fragmented streams of data and atomic messages, and formalized the notion of stream-based secure channels [54]; hierarchies of how channel AEAD notions relate [33]; and multi-key security [59], enabling the analysis of secure channel protocols such as TLS 1.3 [102]. The TLS protocol has in fact played a key role in several works, motivating a better understanding of secure channels, with related works spanning robustness [55], channel termination [31], and alternative channel security notions [14]. However, these existing constructions were designed to model secure channels in the context of the terrestrial Internet and are therefore not readily adaptable to address the unique nuances of BPSec.

While BPSec is a secure channel that focuses on the confidentiality and integrity of the data transmitted the unique nature of its construction hinders capturing its security within

any existing model for secure channels. A typical BPSec state simultaneously maintains a shared state with multiple parties, which include the source and destination of a bundle, as well as any honest intermediate node that might process the bundle on its way towards the final destination. This behavior is distinctly different to a secure channel on the terrestrial Internet, where communications are end-to-end protected, and only the sending and receiving parties at either end of the channel can process the messages. Moreover, unlike a typical message transmitted within a secure channel, the length of a BPSec bundle may be constantly changing, which is an inherent part of its construction. Intermediate nodes that process a bundle may add or remove layers of security from a bundle as per their local policies for processing. This layering of security may sound similar to the design of Tor onion routing [120] but we highlight that these two protocols are strikingly dissimilar for various reasons; unlike a Tor circuit, BPSec nodes do not know their predecessor and successor; message re-encryption is strictly prohibited in BPSec; BPSec integrity checking is performed on a hop-by-hop basis and is not necessarily end-to-end and; anonymity is not a property captured within the BPSec design. Furthermore, BPSec allows for the partial processing of bundles, i.e., intermediaries may only process blocks from a bundle for which they share a state with the respective source of that block, which may be the bundle source or another intermediate node. This rather "flexible" secure channel construction of BPSec cannot be successfully captured within the rigid formalisms of any existing secure channel frameworks. Thus, we introduce a novel Flexible Secure Channels (FSC) security definition in Section 4.3 that is capable of capturing the unique nature of security goals for the BPSec protocol. Our formalism for FSC provides a framework to model secure channels in a "group" setting, which we have tailored to capture a multi-user and multi-ciphersuite use case with diverse corruption parameters. We note that our formalism is "flexible" both in the sense that it captures the fluid nature of BPSec security but can also be easily generalized to analyze the security of any channel protocol.

4.2 BPSec Formalization and StrongBPSec

We first present a formalization of notation for BPSec, with variables shown in Table 4.2 and we further highlight all cryptographically relevant fields within a general BPSec block in Figure 4.7.

As described in 4.1.3 a *Bundle Protocol Agent* (BPA) could play three separate roles within a delay tolerant infrastructure: a *Source Node* (SN) that originates a bundle, an

$\vec{\Pi}$	Global state $\{st_1, ..., st_n\}$ that holds particular key and parameter sets $\{st_i[p] = (\vec{k}_p, p)\}$ used by Party i participating in a BPSec channel
aad	Authenticated associated data
bid	Block index which identifies the block written as a subscript of $\vec{\mathtt{M}}$, $\vec{\mathtt{F}}$, or $\vec{\mathtt{P}}$: B - Index of the Strong BPSec ledger block $\mathtt{BIB}_\mathtt{B}$ (always the last block in our construction, but can be placed before PLD to align with [35]) PB - Index of the Primary Block (always 0) PLD - Index of the bundle payload block (always $\mathtt{B}-1$) tar - Index of security operation's target block
$\vec{\mathtt{C}}$	Resultant array after operating on $\vec{\mathtt{M}}$ with a set of cryptographic operations specified by $\vec{\mathtt{F}}$ and parameterized by $\vec{\mathtt{P}}$.
c_k	Ciphertext of the key wrapped ephemeral symmetric key k
ctr	Counter for tracking index of security blocks
$\vec{\mathtt{F}}$	List of security operations $\{\mathsf{conf}, \mathsf{int}\}$ for blocks in a bundle $\vec{\mathtt{M}}$
$\vec{\mathtt{F}}$.type	BPSec block type {"BCB", "BIB"} synonymous with operations $\{\mathsf{conf}, \mathsf{int}\}$
$\vec{\mathtt{F}}$.targets	Security operation targets $\{\mathsf{bid}_0, \ldots, \mathsf{bid}_\mathsf{N}\}$
$\vec{\mathtt{ID}}$	A global set of node identifiers consisting of $id \in \{id_1, \ldots, id_n\}$ to uniquely identify each state st in global state $\vec{\Pi}$
IV	Initialization vector
k	Ephemeral symmetric key (e.g. the key wrapped key)
k_p	Preshared symmetric key accessible to parties with access to $\vec{\Pi}$
$k^{\mathtt{BB}}$	Ephemeral symmetric key for $\mathsf{BIB}_\mathtt{B}$ authentication (i.e. the key wrapped $\mathsf{BIB}_\mathtt{B}$ key)
$k_p^{\mathtt{BB}}$	Preshared symmetric key for $\mathsf{BIB}_\mathtt{B}$ authentication
$k^{\mathtt{RR}}$	Ephemeral symmetric key for BRB authentication (i.e. the key wrapped BRB key)
$k_p^{\mathtt{RR}}$	Preshared symmetric key for BRB authentication
$\vec{\mathtt{M}}$	Plaintext blundle made of blocks $\{\mathsf{bid}_0, \ldots, \mathsf{bid}_\mathsf{n}\}$ where $\mathsf{n} = \lvert\vec{\mathtt{M}}\rvert$ such that $\lvert\vec{\mathtt{M}}\rvert \geq 1$ (mandatory primary and payload blocks)
$\vec{\mathtt{M}}_i.m$	The plaintext data in a block $\vec{\mathtt{M}}_i$
meta	Metadata ledger for all security operations performed by the security source that is authenticated by the $\mathtt{BIB}_\mathtt{B}$
$\vec{\mathtt{P}}$	Sets of ciphersuite parameters associated with security operations $\vec{\mathtt{F}}$ specified by a security context.
$\vec{\mathtt{P}}$.alg	Algorithms specified (e.g. AEAD modes, key generation) for a block.
$\vec{\mathtt{P}}.id$	Node identifier for a party that supports a particular parameter set
$\vec{\mathtt{P}}$.params	Security parameters used for particular algorithms.
$\vec{\mathtt{P}}$.params.kid	Key identifier
$\vec{\mathtt{P}}$.init	T/F flag: T if the security source is the bundle source; F otherwise.
st	The local state of a party within global state $\vec{\Pi}$
$st.id$	The local state for party id
tar	The target block index: $\vec{\mathtt{M}}_{\mathtt{tar}}$ is the unprotected target block whereas $\vec{\mathtt{C}}_{\mathtt{tar}}$ is the protected target block.
valid	A helper function to check if the security operation $\vec{\mathtt{F}}_i$ conflicts with any existing security operations already applied to a block $\vec{\mathtt{M}}_i$.

Table 4.2: Abstracted variables used to formalize BPSec.

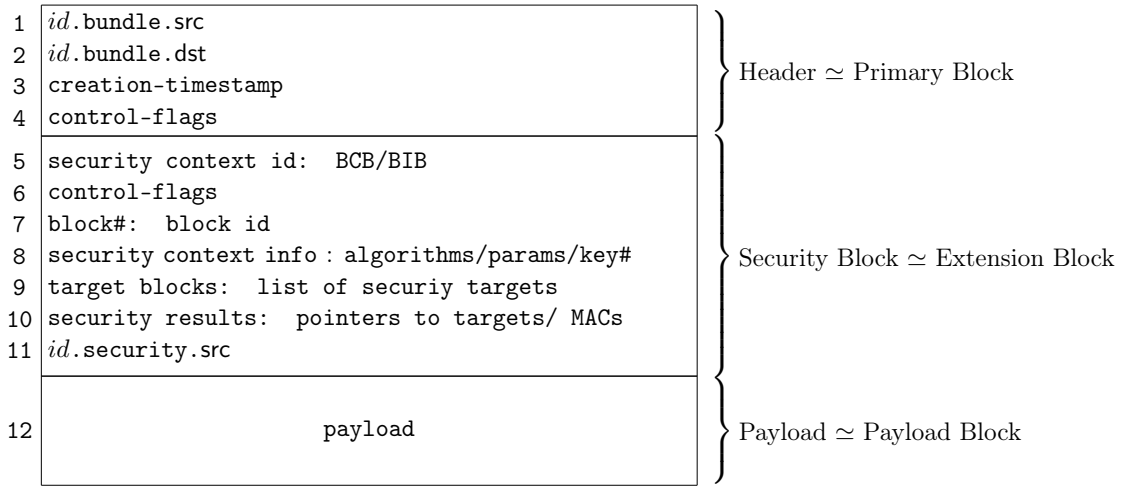


Figure 4.7: Generic BPSec bundle with cryptographically relevant fields.

Intermediate Nodes (IN) that receive and process bundles— potentially adding or discarding component blocks before forward transmission or a *Destination Node* (DN) that receives a bundle. We assume a 1:1:1 relationship between $\overrightarrow{\texttt{M}}, \overrightarrow{\texttt{F}}, \overrightarrow{\texttt{P}}$ meaning that for each block in a bundle $\overrightarrow{\texttt{M}}$, a BPA $\in$ {SN, IN} can add a unique security operation $\overrightarrow{\texttt{F}} \in$ {"BIB", "BCB"} governed by a unique cryptographic parameter set $\overrightarrow{\texttt{P}}$. The security policy of a BPA may be configured to ignore certain blocks which corresponds to a particular $\overrightarrow{\texttt{F}}_i = \perp$. Otherwise, the BPA either adds plaintext integrity or authenticated encryption operations to the block yielding in a new BIB or BCB, respectively. We designate the BPSec protected bundle as $\overrightarrow{\texttt{C}}$. For authenticity protection, the BPA performs HMAC operations using a secret key to the block and stores the tag, a security result, in the BIB. For authenticated encryption, AEAD is applied to the block yielding either separate MAC and ciphertext or single combined result: in cases where the MAC is generated separately, it can be stored in the BCB as a result while the ciphertext replaces the data being encrypted in-place. When multiple blocks are protected by the same security operation using the same parameter set, they are consolidated under a single BIB or BCB respectively as seen in Figure 4.5.

4.3 Flexible Secure Channels

In this section we formalize our notion of *flexible secure channel* and its security. In Section 4.6 we leverage our *flexible secure channels* formalism to analyze the security of the BPSec protocol and our StrongBPSec construction.

A Flexible Secure Channel (FSC) is distinguished by several features from a traditional secure channel. Specifically, an FSC includes not only a sender and receiver but also one or more intermediate nodes which share a common set of keys $\overrightarrow{k}$ and associated parameters $p \in \overrightarrow{\mathtt{P}}$ contained in their individual *states*. For BPSec, the keys are preinstalled symmetric keys and the parameters are dictated by the security context(s) supported by the participant. The FSC global state $\overrightarrow{\Pi}$ is the union of all local states, st of FSC participants (i.e. $st \in \overrightarrow{\Pi}$). Without loss of generality, in a symmetric key FSC, such as in BPSec under the Default Security Context, sender state st and receiver state st' must match (i.e. there exists $p \in \overrightarrow{\mathtt{P}}$ such that $st[p] = st'[p]$) to correctly process FSC messages.

Definition 35 (Flexible Secure Channels)**.** *A flexible secure channel* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv})$ *with associated state space* $\mathcal{S}$ *and error space* $\mathcal{E}$*, where* $\mathcal{E} \cap \{0,1\}^* = \emptyset$*, consists of three efficient algorithms:*

- Init. *On input of a security parameter array* $\overrightarrow{\mathtt{P}}$*, this probabilistic algorithm outputs initial state array* $\overrightarrow{\Pi} \in (\mathcal{S} \times \ldots \times \mathcal{S})$*. We write* $(\overrightarrow{\Pi}) \xleftarrow{\$} \mathsf{Ch.Init}(\overrightarrow{\mathtt{P}})$*. We note that global state* $\overrightarrow{\Pi}$ *constitute many states* st*, where a state* st_s *matches with multiple other states* $st' \in \{st_1, \ldots, st_n\}$*.*
- Snd. *On input of state* $st \in \overrightarrow{\Pi}$*, a message bundle* $\overrightarrow{\mathtt{M}} : |\overrightarrow{\mathtt{M}}| \in \mathbb{N}$ *and* $\forall m \in \overrightarrow{\mathtt{M}}, m \in \{0,1\}^*$*, a security flag array* $\overrightarrow{\mathtt{F}}$*, and a security parameter array* $\overrightarrow{\mathtt{P}}$*, this (possibly) probabilistic algorithm outputs an updated state array* $st' \in \overrightarrow{\Pi}$ *and a secured bundle* $\overrightarrow{\mathtt{C}} : |\overrightarrow{\mathtt{C}}| \in \mathbb{N}$ *and* $\forall c \in \overrightarrow{\mathtt{C}}, c \in \{0,1\}^*$*. We write* $(st', \overrightarrow{\mathtt{C}} \xleftarrow{\$} \mathsf{Snd}(st, \overrightarrow{\mathtt{M}}, \overrightarrow{\mathtt{F}}, \overrightarrow{\mathtt{P}}))$*.*
- Rcv. *On input of a state* $st \in \overrightarrow{\Pi}$ *and a secured bundle* $\overrightarrow{\mathtt{C}}$*, this deterministic algorithm outputs an updated state* $st' \in \overrightarrow{\Pi}$ *and a message bundle* $\overrightarrow{\mathtt{M}} : \forall\ m \in \overrightarrow{\mathtt{M}}$ *where* $m \in (\{0,1\}^* \cup \mathcal{E})$*. We write* $(st', \overrightarrow{\mathtt{M}} \leftarrow \mathsf{Rcv}(st, \overrightarrow{\mathtt{C}}))$*.*

Remark 2. *We note that our flexible secure channel* FSC *design perfectly models the peculiarities of the BPSec construction. Our* FSC.Snd *algorithm models the behavior of an initial BPSec source node that constructs and sends a bundle* $\overrightarrow{\mathtt{C}}$*, as well as the behavior of an intermediate node that modifies and forwards the same* $\overrightarrow{\mathtt{C}}$*. Similarly,* FSC.Rcv *algorithm*

models the behavior of an intermediate node that partially processes a bundle $\overrightarrow{\mathtt{C}}$*, as well as the behavior of the destination node that completely processes the bundle* $\overrightarrow{\mathtt{C}}$ *to extract* $\overrightarrow{\mathtt{M}}$*.*

Definition 36 (Flexible secure channel correctness)**.** *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv})$ *be a flexible secure channel. We say that* Ch *provides correctness if for all state pairs* $(st_i, st_j) \in \overrightarrow{\Pi} \xleftarrow{\$} \mathsf{Ch.Init}(\overrightarrow{\mathtt{P}})$*, for all messages* $(m_0, \ldots, m_\ell) : \overrightarrow{\mathtt{M}} = \{m_0, \ldots, m_\ell\}$ *(where* $\ell \in \mathbb{N}$*) , for all flags* $(f_0, \ldots, f_\ell) : \overrightarrow{\mathtt{F}} = \{f_0, \ldots, f_\ell\}$*, for all parameters* $(p_0, \ldots, p_\ell) : \overrightarrow{\mathtt{P}} = \{p_0, \ldots, p_\ell\}$ *and for all message arrays* $\overrightarrow{\mathtt{M}}' \leftarrow \mathsf{Rcv}(st_j, \mathsf{Snd}(st_i, \overrightarrow{\mathtt{M}}, \overrightarrow{\mathtt{F}}, \overrightarrow{\mathtt{P}}))$*, we have that*

$$(m_0, \ldots, m_\ell) \in \overrightarrow{\mathtt{M}}' \iff \forall p_r \in \{p_0, \ldots, p_\ell\} : i, j \in p_r.id \quad .$$

For a block to be processed correctly by a receiver, the receiver must support the particular parameter set embedded in the associated states for each of the message blocks in the bundle. Additionally, since an intermediate receiving node may not share a matching state (with appropriate parameter sets) for all security blocks in a bundle $\overrightarrow{\mathtt{C}}$, the node may legitimately process only the blocks for which it shares the correct state. This *partial processing* is additionally captured as correct behavior via our definition.

Remark 3. *We draw the reader's attention to the difficulty in defining* FSC *correctness. This is due to the modular nature of BPSec bundles which may have different sets of parameters* $p \in \overrightarrow{\mathtt{P}}$ *associated with each security block* $m \in \overrightarrow{\mathtt{M}}$*. Thus, for a receiver to correctly process each* $m \in \overrightarrow{\mathtt{M}}$*, they must also support corresponding security parameters* $p \in \overrightarrow{\mathtt{P}}$ *per each block they process.*

In our protocol construction for BPSec and StrongBPSec illustrated in Figures 4.11 (Init algorithm), 4.12 (Snd algorithm) and 4.13 (Rcv algorithm) we leverage the flexibility and modularity of our formalism for flexible secure channels. Notably, within the BPSec construction, the intermediary nodes process bundles differently to source and destination nodes, who only Snd and Rcv bundles respectively. In contrast, intermediary nodes both Snd and Rcv bundles; they Snd as they add new BCB/BIB blocks to a bundle and forward it to the next node; they Rcv bundles forwarded to them within which they process and discard BIB/BCB blocks and add BRB read receipts. The versatility of our flexible channel design successfully captures all these use cases, enabling the formalization of uncommon constructions such as BPSec. Figure 4.8 illustrates an execution of BPSec as a FSC.

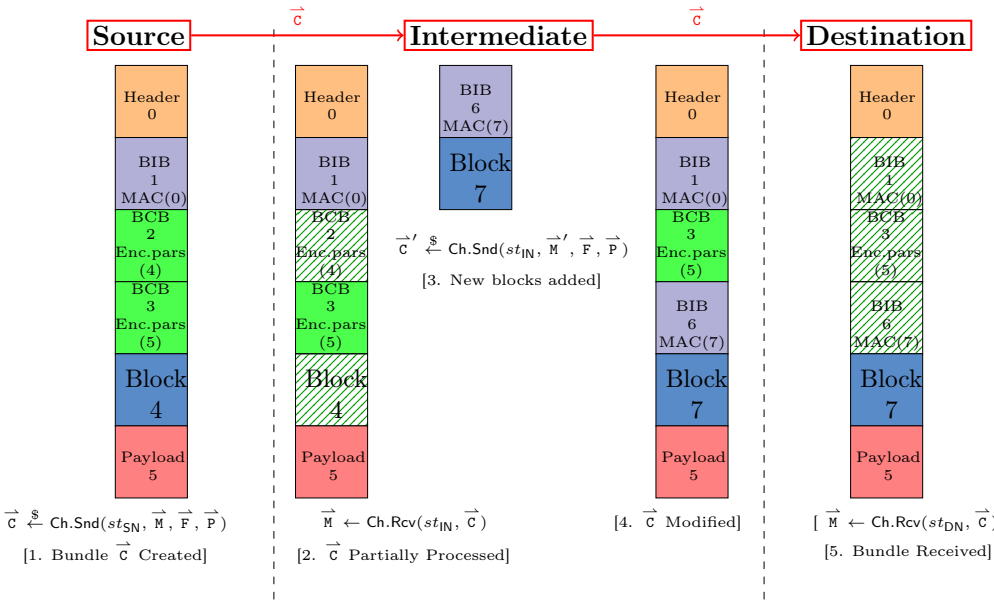


Figure 4.8: BPSec as a FSC*. All processed and discarded blocks are represented in green diagonal lines. The values for* $\overrightarrow{\mathtt{P}}$ *and* $\overrightarrow{\mathtt{F}}$ *are selected from respective sets which can be found in Table 4.2.*

4.3.1 BPSec Security Challenges & Threat Model

We note that the DTN threat model gives an attacker complete network access, affording them read/write access to bundles traversing the network. Eavesdropping, modification, topological, and injection attacks are all described in [28, Section 8]. Therein, these "on-path attackers" can be unprivileged, legitimate, or privileged nodes depending on their access to cryptographic material: unpriviledged nodes only have access to publicly shared information, legitimate nodes have additional access to keys provisioned for itself, and privileged nodes have further access to keys (privately) provisioned for others. Our symmetric key-based security context, detailed in Section 4.5, builds upon the BPSec Default Security Context [29]. While the BPSec Default Security Context assumes all BPSec participants are privileged and recognizes that attackers may be either privileged or unprivileged, it does not provide security guarantees against privileged attacks.

In an effort to distinguish malice by INs, we further abstract these classes into *honest* and *dishonest* nodes in our analysis. Honest INs are priviledged nodes that faithfully execute the role of a BPA as described in Section 4.1.3. Dishonest INs are priviledged nodes that

attempt to violate the integrity or confidentiality of blocks it processes (e.g. by dropping or modifying blocks), and captured by our adversary model. We observe a specific gap in guarantees BPSec provides through the BIB, BCB, and the default security context. Specifically, BPSec has no cryptographic auditing mechanism for detecting unpriviledged modifications to a bundle between the SN and DN.

We also highlight that, due to the intrinsic design of BPSec, which permits *privileged* intermediate nodes to process and discard bundles before they reach their intended destination node, it is not feasible to directly align the notion of end-to-end security in the conventional sense to the non-terrestrial DTN paradigm. Accordingly, we define our adversary as an *internal, privileged-but-dishonest* intermediary node attempting to compromise the integrity or confidentiality of the blocks it processes. BPSec, in its current instantiation, does not defend against *external, unprivileged* adversaries who indiscriminately drop bundles to induce denial-of-service; such attacks are therefore considered out of scope for this work.

Claim: BPSec protections against plaintext modification are insufficient and can lead to a self-imposed denial-of-service. A similar argument can be made for authenticated encrypted BCB target blocks.

- Suppose an unpriviledged dishonest IN strips the BIB(s) and/or BCBs from all bundles it receives and forwards and appends an additional bit to all their associated security targets. BPSec has no in-band mechanisms (i.e. cryptographically enforced) to detect or correct this kind of modification (aside from encrypting the BIB targets after they are authenticated). According to [28, 5.1.2], other BPAs who receive this stripped bundle will use an out-of-band security policy mechanism to determine whether to drop, modify, or forward the bundle. Under the recommended security policy, BPAs will remove all target blocks that were supposed to be protected by a BIB. This could lead to dropping the entire bundle if the security policy specified that the primary block must be BIB protected.
- This paradigm sacrifices availability over authenticity. One could argue that availability was never guaranteed by BP and that it is out of scope of BPSec to prioritize availability over authenticity (resp. confidentiality) [2].

[2]We also note that, if only a basic integrity check – not authenticity – of the data is required, the use of the BIB to provide integrity protection is unnecessary as cyclic redundancy check (CRC) codes already exist in BP blocks. CRCs do not provide authenticity.

- Under a unpriviledged dishonest attacker model the DN would not have a means to detect dropped target blocks. This may lead to the DN incorrectly assuming that they have a complete message and acting on it, even if core actionable information was in the dropped blocks.

The core impact of the issues highlighted above is that the destination BPA is unaware of what messages have been removed by the intermediate nodes. To address this, we provide a strong BPSec variant StrongBPSec that offers *read receipts*. Read receipts are designed to provide a verifiable record for processed blocks, ensuring transparency while adding an additional layer of integrity protection between SN and DN within BPSec's symmetric key constraints. Our read receipt design facilitates the maintenance of a *ledger* block that guarantees the integrity of the original bundle created by SN, while simultaneously permitting honest intermediaries to process and discard SN-created security blocks. This should not be construed with the "return-receipt" from a Bundle Status Report [126] which is a out-of-band (i.e. separate from the bundle) policy-driven acknowledgment of receipt or change status. In fact, an on-path-attacker can simply drop all Bundle Status Reports whereas StrongBPSec offers a stronger in-band (i.e. to the bundle itself) cryptographically enforced audit log. We begin by offering an intuition for StrongBPSec, followed by a formalization of its construction alongside the standard BPSec protocol. Figures 4.11, 4.12, and 4.13 provide a side-by-side comparison to clarify the differences between the two approaches.

4.4 StrongBPSec with Read Receipts

As noted, the current BPsec specification [28] provides no security guarantees against arbitrary message dropping by an attacker during the transit of a bundle between its source and the final destination. Particularly, an inherent feature of BPSec is to allow honest intermediary nodes to process and discard bundle blocks as governed by their internal policies with the exception of primary and payload blocks. However, BPSec [28] does not require intermediate nodes to provide verifiable evidence of identity in order to process or discard security blocks. Neither does the bundle maintain a record of changes made to it on way to its destination. Thus, an external attacker can arbitrarily and selectively drop blocks from a bundle en route without being detected. The proposed COSE Security Context for BPSec [114] described in Section 4.1.4, as part of its integration into BPSec, includes a recommendation to increase the integrity of bundle blocks. In order to provide stronger

integrity guarantees, they propose binding an arbitrary number of blocks together in the same bundle, using additional associated data (`aad`) for each block concerned. However, we argue that binding `aad` to an arbitrary number of blocks in this manner also restricts the ability of an honest intermediary to process and discard any individual block from such a group. In an attempt to find a fair middle ground between the somewhat relaxed original functionality goals of BPSec [28] and the overly rigid security guarantees of COSE[114], we propose our StrongBPSec protocol with *read receipts*.

StrongBPSec introduces two additional layers of integrity that aims to guarantee to the bundle destination one of two of the following:

1. All blocks added by the bundle source node SN has arrived unchanged at the bundle destination node DN, or
2. Some honest intermediary node IN, who supports and has access to appropriate cryptographic parameters, has correctly processed and discarded block/s that were constructed by the bundle source SN.

Note that, for clarity, we will be exclusively distinguishing the roles of BPA as either SN, IN or DN for the rest of this discussion.

We achieve these stronger integrity guarantees through the addition of two additional checks leveraging the already existing BIB format of BPSec. As illustrated in Figure 4.9, we introduce a BIB_B block, calculated and appended to the end of bundle by the original source node SN. As the originator of the bundle, SN constructs BIB_B, which includes a MAC tag calculated over the payload block and a set of uniquely identifying details for each security block added by SN. These identifying details include unique key/block identifiers (kid/bid), and the number of target blocks `len`(BIB/BCB) covered by a security block (BIB/BCB). Once constructed, this BIB_B is only verified by the destination node DN [3]. The successful verification of BIB_B with the corresponding key k^{BB}, which is either a pre-shared or a key-wrapped key, against the received bundle informs DN one of two things; either they have received the exact same bundle SN constructed; or an honest intermediary IN has processed and discarded some of the blocks but have replaced them with correct read receipts BRB. These BRB blocks duplicate the unique identifying details of a discarded block authenticated with MAC tag calculated over this information, which we discuss next. In line with the BPSec specification, we assume only honest privileged intermediary nodes who support

[3] We note that we only consider the use case for DN for simplicity but WLOG we say that any *security acceptor* authorized by the security policy may process BIB_B.

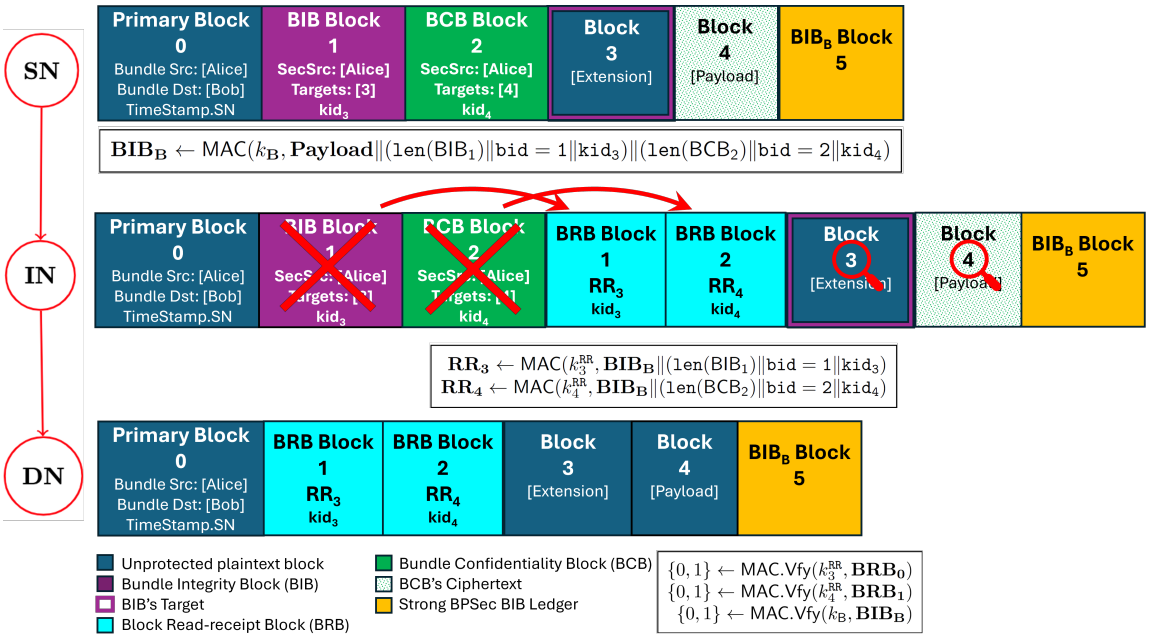


Figure 4.9: StrongBPSec *with Read Receipts. Source integrity block MAC* $\mathsf{BIB_B}$ *is calculated over the payload and a set of identifying details (metadata) for each target/security block (*$\mathsf{BIB_1}$ *and* $\mathsf{BCB_2}$*) added by SN. Read receipts* BRB *are calculated by INs every time they successfully process a security block added by SN, recording and verifying the details of the blocks they process. INs replace any such security blocks they process with a corresponding* BRB *before forwarding the bundle. DN first verifies any* BRB*s added by INs followed by the verification of* $\mathsf{BIB_B}$*.*

and have access to correct cryptographic parameters for each block they process.

The second type of integrity check introduced by our StrongBPSec construction is the concept of a *read receipt* BRB. The addition of BRB allows an honest intermediary acting as a *security acceptor* to process and discard any block added by SN but still maintain a verifiable record of their details that was used in the calculation of $\mathsf{BIB_B}$. To clarify, recall from Section 4.3.1, a read receipt BRB provides an in-band ledger of changes for processing at the DN; it should not be assumed that the BRB gets sent back to the SN as some sort of acknowledgment of receipt or changes as would be done by a "return-receipt" from a Bundle Status Report [126].

A BRB block contains a plaintext and a MAC tag calculated over it. The plaintext duplicates the identifying details of the corresponding discarded security block (kid/bid, len(BIB/BCB)) that was used in the calculation of $\mathsf{BIB_B}$, without which the verification of $\mathsf{BIB_B}$ at DN will fail. A MAC tag is then calculated using a unique key k^{RR}—

which is either a unique pre-shared or a key-wrapped key that is specifically associated with BRB authentication but is independent of the key used for the $\overrightarrow{\mathsf{F}}$ security operations— over the bundle $\mathsf{BIB_B}$ concatenated with this plaintext. These BRB blocks act as a transcript of the original bundle to the DN, who verifies all BRB blocks prior to the verification of $\mathsf{BIB_B}$. The successful verification of BRB blocks within a received bundle informs DN that only honest intermediaries have discarded blocks from the original bundle. In Figure 4.10 we illustrate BPSec within our SFSC formalism, which we describe in detail in Figures 4.11, 4.12 and 4.13 .

Remark 4. *We note that the overhead introduced by adding the new* BRB *and* $\mathsf{BIB_B}$ *block types to the existing bundle structure is minimal and well within the tolerance of DTN architectures. The* $\mathsf{BIB_B}$ *block consists of a single additional* MAC *tag computed over identifying metadata of the original SN-constructed bundle, resulting in negligible overhead. The* BRB *block is appended whenever an honest intermediate node processes and discards a SN-created security block (and potentially its associated target blocks). However, this does not meaningfully increase bundle size or consume significant channel bandwidth, as the* BRB *is itself a compact* MAC *tag computed over duplicated metadata from the discarded security block.*

Admittedly, this adds one additional block per discarded security block compared to the baseline case where an intermediate node simply discards blocks without adding anything. Nonetheless, within DTN architectures, intermediate nodes routinely insert blocks to support routing or facilitate the delivery of bundles towards their destination. Consequently, the inclusion of a BRB *for each processed SN-constructed security block is fully aligned with the operational semantics of BPSec and does not interfere with its expected behavior.*

Moreover, DTNs are explicitly engineered for environments characterized by high latency, disruptions, and limited round-trip opportunities. They are inherently capable of accommodating large bundles with additional overhead, leveraging robust data links and extended local storage to preclude the need for round trips and to enable reliable forwarding once connectivity resumes. As such, these systems are equipped to handle moderate increases in bundle size, making our design choices well-aligned with the operational capabilities and resilience goals of DTN deployments.

Remark 5. *While significant security gains could be achieved with inclusion of digital signatures in this improved protocol (in particular preventing impersonation within a group sharing the same parameter set and keys), the BPSec infrastructure does not assume asymmetric key management; pre-shared symmetric keys were favored for simplicity and broad*

implementation within the constrained deep space environment. Thus, in strengthening BPSec, we inherit the original key infrastructure and primitives assumed by the Default Security Context described in RFC9173 [29]. Therefore, it is not possible to detect which IN has edited the bundle nor is it possible to protect against an insider threat. These issues are inherent to RFC9173 in absence of asymmetric key management.

Figures 4.11, 4.12 and 4.13 respectively illustrate the formal construction for Init, Snd and Rcv algorithms of BPSec under RFC9173 alongside our StrongBPSec protocol. The construction abstracts details of BPSec [28] and its Default Security Context [29] into a FSC protocol (see Definition 35) with modifications needed for a StrongBPSec protocol. We formally prove the respective security of BPSec and StrongBPSec in Section 4.6. Next, we explain the intuition behind our formal construction.

- Init generates symmetric keys for matching BPSec states per parameter set. Parameters in a state are associated with a collection of party identifiers $\vec{id}$ who all maintain the same symmetric keys. Init generates three keys per parameter set: a symmetric key k_p; a read receipt key k_p^{RR} and; a bundle verifier key k_p^{BB}.
- Snd creates security blocks either by bundle source or any intermediate node. For each message $m \in \overrightarrow{\mathtt{M}}$, Snd checks that the security operation is valid, and key wraps a new symmetric key if necessary. Snd adds the appropriate processing information to the security block, and either authenticates or encrypts the message m according to the security operation. In StrongBPSec, Snd also adds a meta-data array for its ledger block, which it authenticates, creating a verifiable ledger.
- Rcv processes security blocks either by bundle destination or any intermediate node. For each ciphertext $c \in \overrightarrow{\mathtt{C}}$, Rcv checks that the security acceptor has the correct keys to process the ciphertext, and key-wraps a new read-receipt key if necessary as intermediate node. Rcv adds read-receipts after processing the security block, and authenticates the associated meta-data. In StrongBPSec, the destination node also verifies all read-receipts, and constructs a meta-data array for the ledger block and verifies the $\mathsf{BIB}_{\mathsf{B}}$, rejecting the payload if any checks fail. Relevant keys are either extracted from state or decrypted from *keywrapped* ciphertexts.

Additionally, in Figures 4.11, 4.12 and 4.13, $X \xleftarrow{\parallel} Y$ denotes concatenating Y to X; $T[j]$ denotes the value accessed by key j in table T, which we assign no particular structure; subscripts are generally used for array indices or but are also used to associate variables (e.g.

k_p to denote a key k associated with p) or to configure a function call (e.g. $\mathsf{Auth}_{\vec{\mathtt{C}}_{i.\mathtt{params}}}$); superscripts are used to classify a variable type (e.g. k^{RR} refers to a read-receipt key); $p.\mathtt{alg}.\mathsf{Gen}$ refers to invoking a key generation function from a family of algorithms such as Key-Wrap (KW) in p but we also overload `alg` as an identifier (e.g. used for boolean comparison).

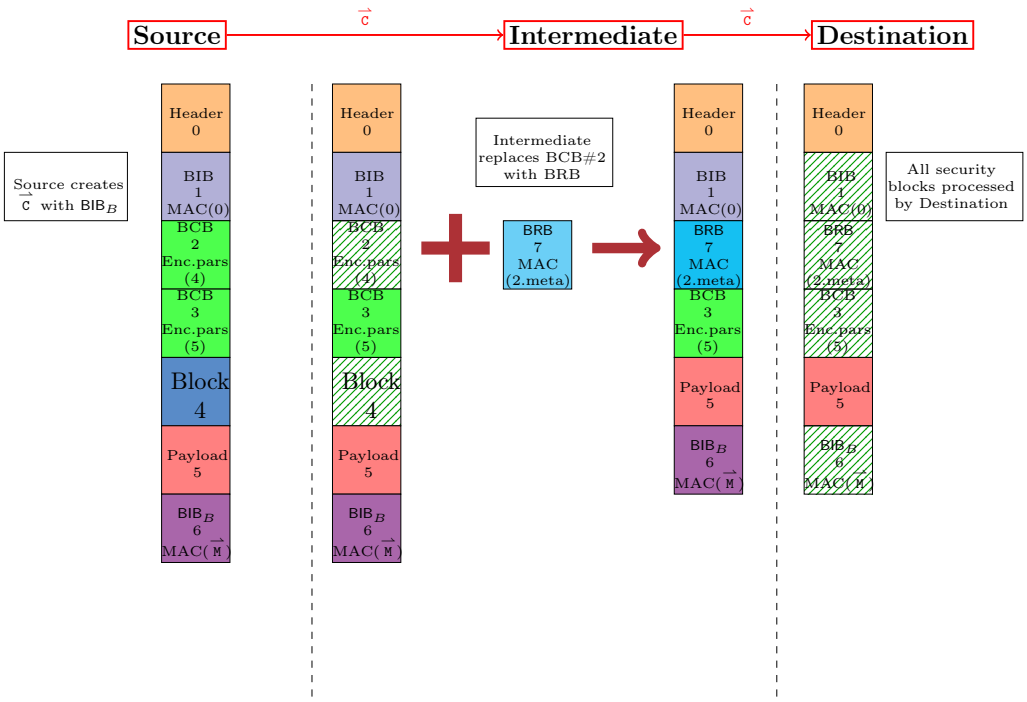


Figure 4.10: Flow of interactions during an expected execution of StrongBPSec. *Source creates a bundle and forwards it to Intermediate. Intermediate processes and replaces* BCB *block #1 with read receipt block* BRB *#7. Destination processes and discards all remaining security blocks and extracts the bundle message* $\vec{\mathtt{M}}$. *All processed and discarded blocks are represented in green diagonal lines. The values for* $\vec{\mathtt{P}}$ *and* $\vec{\mathtt{F}}$ *are selected from respective sets which can be found in Table 4.2.*

4.5 The Flexible and Strong Flexible Secure Channel Models

In Figure 4.14 we formalize the security experiment for Flexible Secure Channels (FSC) that captures the BPSec security goals described by [28]. Furthermore, in Figure 4.14 we also capture the intended security goals added by StrongBPSec, namely through Strong

FSC Security (SFSC) security, highlighting these as additional steps.

$\underline{\mathsf{Init}(\overrightarrow{\mathtt{P}})}$

1: **for** $id \in \overrightarrow{\mathtt{ID}}$ **do**:
2: $st.id \leftarrow \underline{id}$
3: **for** $p \in \mathtt{P}$ **do** :
4: **if** $id \in p.\overrightarrow{id}$: **then**
5: **if** $(\exists id' : st_{id'} \in \overrightarrow{\Pi}) \wedge (id' \in p.\overrightarrow{id})$ **then**
6: $k_p, k_p^{\mathtt{RR}}, k_p^{\mathtt{BB}} \leftarrow st_{id'}[p]$
7: **else**
8: $k_p, k_p^{\mathtt{RR}}, k_p^{\mathtt{BB}} \leftarrow p.\mathtt{alg}.\mathsf{Gen}(p.\mathtt{params})$
9: **end if**
10: $st_{id}[p] \leftarrow ((k_p, k_p^{\mathtt{RR}}, k_p^{\mathtt{BB}}), p)$
11: **end if**
12: **end for**
13: $\overrightarrow{\Pi} \xleftarrow{\cup} st_{id}$
14: **end for**
15: **return** $\overrightarrow{\Pi}$

Figure 4.11: BPSec protocol Init *algorithm construction. Refer to Table 4.2 for notational definitions.*

In the FSC experiment, the challenger begins by generating the full secret state for all parties, and randomly sampling a bit b. The adversary is then split into two phases. First, in the Corrupt phase, the $\mathcal{A}$ is allowed to issue Corrupt queries for particular parameter sets[4]. In the next phase, $\mathcal{A}$ outputs some state, which is given as input to the next adversary $\mathcal{A}'$, which allows us to capture static corruption adversaries.

We argue that proving the security of our construction against a static (as opposed to adaptive) $\mathcal{A}$ is sufficient for the following reasons: each BPSec block is associated with a unique set of independent security parameters that remain unchanged for the duration of the message lifetime; there is no forward secrecy and it does not make a difference at what point of a message's lifetime $\mathcal{A}$ compromises these security parameters as they do not update; and adversary $\mathcal{A}$ compromising of set of security parameters for a specific block does not affect the security of any other blocks in the same bundle.

$\mathcal{A}$ now has access to two oracle queries: $\mathcal{O}$Snd and $\mathcal{O}$Rcv:

[4]This describes how $\mathcal{A}$ becomes a dishonest privileged node.

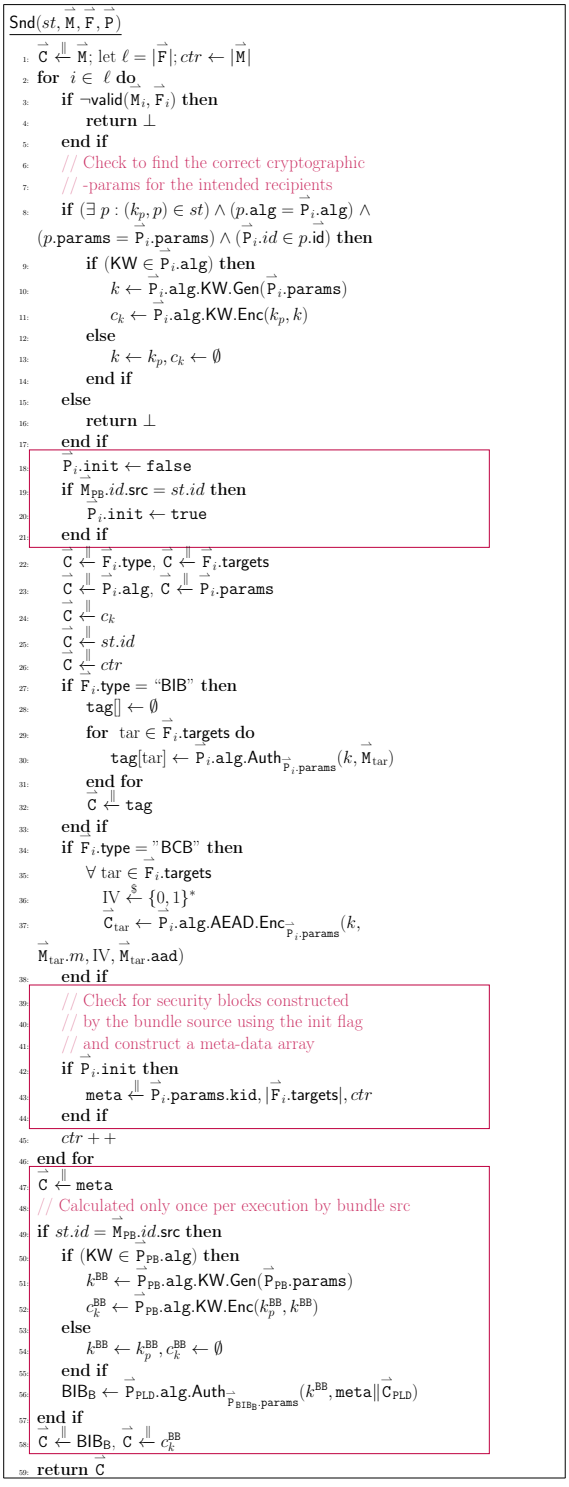


Figure 4.12: BPSec protocol Snd *algorithm construction. The additions needed for the* StrongBPSec *protocol is highlight in boxes. Refer to Table 4.2 for notational definitions.*

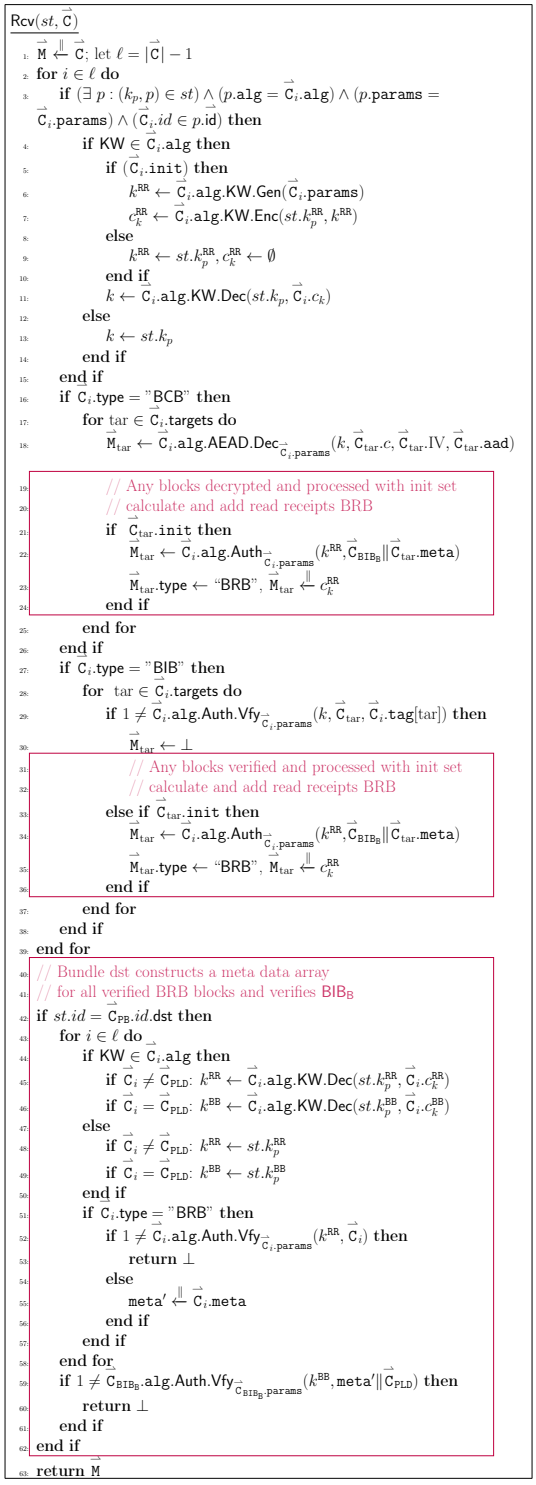


Figure 4.13: BPSec protocol Rcv *algorithm construction. The additions needed for the* StrongBPSec *protocol is highlight in boxes. Refer to Table 4.2 for notational definitions.*

- $\mathcal{O}\mathsf{Snd}(i, \vec{\mathtt{M}}, \vec{\mathtt{F}}, \vec{\mathtt{P}}) \rightarrow \vec{\mathtt{C}}$: allows the adversary to indicate that party i should protect the plaintext bundle $\vec{\mathtt{M}}$ using security operations $\vec{\mathtt{F}}$ using security parameters $\vec{\mathtt{P}}$. If the bit b sampled by the challenger is 1, the associated parameters have not been corrupted, and the security operation is encryption (signified by $\mathsf{type} = \mathsf{BCB}$) then the challenger records and replaces the associated ciphertext with random strings from the same length. A register CTXT is maintained to record these changes to BCB ciphertexts. $\mathcal{O}\mathsf{Snd}$ also records all integrity protected blocks (signified by $\mathsf{type} = \mathsf{BIB}$), and their associated tags in the AUTH register (to detect integrity wins later in the experiment).
- $\mathcal{O}\mathsf{Rcv}(i, \vec{\mathtt{C}}) \rightarrow \vec{\mathtt{M}}$: allows the adversary to indicate that party i should process the ciphertext $\vec{\mathtt{C}}$ recorded in register CTXT. $\mathcal{O}\mathsf{Rcv}$ also checks if the adversary has caused any win events to trigger. Specifically, if the associated security parameters are not corrupted, and the party i outputs a valid block which no honest party produced, then the adversary has forged this block, and the challenger sets $\mathtt{win} \leftarrow \mathtt{true}$.

At some point the adversary terminates and outputs a bit b', and the output of the experiment is $(b = b') \vee \mathtt{win}$. We say an adversary $\mathcal{A}$ wins the FSC security game if they succeed in achieving one of the following: forges an integrity-protected block; forges an authenticated ciphertext within a block or; distinguishes the real-or-random ciphertext within a block. Below we briefly summarize additional notations used in our security experiment illustrated in Figure 4.14 for our security experiment.

- AUTH: A register for honestly authenticated messages and their MAC tags.
- CORR: A register used to track corrupted parameters for determining wins.
- CTXT: A register that maintains either real ciphertext and associated parameter pairs or uniformly random strings from the same space.
- Extr: A helper function to extract a parameter set from a block and state. Extr abstracts away the initial checks performed by a node to confirm whether it supports the correct cryptographic parameters for a given block.
- Proc: A binary function that checks if a ciphertext bundle $\vec{\mathtt{C}}$ can be processed or not. Proc abstracts away the out-of-band policy checks within BPSec.
- tsid: A state identifier that uniquely identifies the parameter set used by bundle source. This is equivalent to the timestamp added by a bundle source within BPSec when bundle $\vec{\mathtt{C}}$ is first constructed.

- TXT:A register for all plaintext messages accessible by the global state $\overrightarrow{\Pi}$.

We now formally define FSC security.

Definition 37 (FSC Security)**.** *Let* Ch *be a channel protocol* $\mathsf{Ch} = \{\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv}\}$. *Let* $\mathsf{Exp}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *be the* FSC *security experiment without red-boxed lines defined in Figure 4.14. We define the advantage* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *that the adversary* $\mathcal{A}$ *wins the* FSC *game as* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = |2 \cdot \Pr(\mathsf{Exp}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = 1) - 1|$. *We say that* Ch *is* FSC*-secure if* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ.

Next, we turn to defining the Strong FSC game (SFSC), which adds the boxes in Figure 4.14. The SFSC game adds two additional win conditions in the Rcv query, allowing the adversary to win by dropping $\overrightarrow{\mathtt{C}}$ blocks without being detected, or by forging so-called read receipts, which indicate to the receiving party that an honest party has "processed" a block from the sender. As before, the adversary at some point will terminate and outputs a bit b' and the output of the experiment is $(b = b') \vee \mathtt{win}$. Thus, in addition to the FSC win conditions, the SFSC adversary $\mathcal{A}$ wins if it can forge either BRB or $\mathsf{BIB}_{\mathsf{B}}$ with non-negligible probability. We now formally define SFSC security.

Definition 38 (SFSC Security)**.** *Let* Ch *be a channel protocol* $\mathsf{Ch} = \{\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv}\}$. *Let* $\mathsf{Exp}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *be the* SFSC *security experiment defined in Figure 4.14 (with all lines included). We define the advantage* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *that the adversary* $\mathcal{A}$ *wins the* SFSC *game as* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = |2 \cdot \Pr(\mathsf{Exp}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = 1) - 1|$. *We say that* Ch *is* SFSC*-secure if* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ.

4.6 Security Analysis

In this section, we provide a security analysis of the BPSec construction given in Figures 4.11, 4.12 and 4.13 under Flexible Secure Channel security described in Definition 37. Next, we provide a security analysis of the $\mathsf{StrongBPSec}$ construction given in Figure 4.11, 4.12 and 4.13 under Strong FSC security in Definition 38. We begin with our analysis of BPSec.

Theorem 6 (FSC Security for BPSec)**.** *Let* n_a *be the total number of parameter sets used in the experiment, and let* n_m *be the total number of key-wraps keys in the experiment. The* BPSec *protocol presented in Figures 4.11, 4.12 and 4.13 is* FSC*-secure. That is, for any PPT algorithm* $\mathcal{A}$ *against the* FSC *security experiment (described in Definition 37)* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A}}(\lambda)$

$\underline{\mathsf{Exp}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)}$

$\vec{\Pi} \leftarrow \mathsf{Ch.Init}(\lambda)$; $\mathtt{win} \leftarrow \mathtt{false}$
$b \xleftarrow{\$} \{0,1\}$
$st \leftarrow \mathcal{A}^{\mathsf{Corrupt}}()$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Snd},\mathcal{O}\mathsf{Rcv}}(st)$
return $(b = b') \vee (\mathtt{win})$

$\underline{\mathcal{O}\mathsf{Rcv}(i, \vec{\mathtt{C}}) \rightarrow m}$

for $\ell \in |\vec{\mathtt{C}}|$ **do**
  $p \leftarrow \mathsf{Extr}(\vec{\mathtt{C}}_\ell, st_i)$
  if $(\mathsf{type}(\vec{\mathtt{C}}_\ell) =$ "BCB"$) \wedge (p \notin \mathsf{CORR})$ **then**
    for tar $\in \vec{\mathtt{C}}_\ell.\mathsf{targets}$ **do**
      $(c, \cdot) \leftarrow \mathsf{CTXT}[\vec{\mathtt{C}}_{\mathrm{tar}}]$
      $\vec{\mathtt{C}}_{\mathrm{tar}} \leftarrow c$
    end for
  end if
end for
$q \leftarrow \mathsf{Proc}(\vec{\mathtt{C}})$
$st_i, \vec{\mathtt{M}} \leftarrow \mathsf{Ch.Rcv}(st_i, \vec{\mathtt{C}})$
if $q \wedge (\vec{\mathtt{M}} \neq \perp)$ **then**
  for $\ell \in |\vec{\mathtt{C}}|$ **do**
    $p \leftarrow \mathsf{Extr}(\vec{\mathtt{C}}_\ell, st_i)$
    // BRB forgery register for SFSC
    if $(\vec{\mathtt{C}}_\ell.\mathsf{type} =$ "BRB"$) \wedge (\vec{\mathtt{C}}_{id.\mathsf{src}} = st_i.id) \wedge p \notin \mathsf{CORR}$ **then**
      for tar $\in \vec{\mathtt{C}}_\ell.\mathsf{targets}$ **do**
        $\mathsf{AUTH} \xleftarrow{\cup} (\vec{\mathtt{C}}_{\mathrm{tar}}, \vec{\mathtt{C}}_j.\mathtt{tag}[\mathrm{tar}])$
      end for
    end if
    if $p \notin \mathsf{CORR}$ **then**
      for tar $\in \vec{\mathtt{C}}_\ell.\mathsf{targets}$ **do**
        if $\vec{\mathtt{C}}_\ell.\mathsf{type} =$ "BIB"$\wedge ((\vec{\mathtt{C}}_{\mathrm{tar}}, \vec{\mathtt{C}}_\ell.\mathtt{tag}[\mathrm{tar}]) \notin \mathsf{AUTH})$ **then**
          $\mathtt{win} \xleftarrow{\cup} \mathtt{true}$
        else if $\vec{\mathtt{C}}_\ell.\mathsf{type} =$ "BCB"$\wedge ((\vec{\mathtt{C}}_{\mathrm{tar}}, p) \notin \mathsf{CTXT})$ **then**
          $\mathtt{win} \leftarrow \mathtt{true}$
          // SFSC win condition for BRB forgery
        else if $\vec{\mathtt{C}}_\ell.\mathsf{type} =$ "BRB"$\wedge ((\vec{\mathtt{C}}_{\mathrm{tar}}, \vec{\mathtt{C}}_\ell.\mathtt{tag}[\mathrm{tar}]) \notin \mathsf{AUTH})$ **then**
          $\mathtt{win} \leftarrow \mathtt{true}$
        end if
      end for
    end if
  end for
end if
// SFSC win condition for BIB_B forgery
for $\mathtt{M}_i \in \vec{\mathtt{M}} : ((\mathtt{M}_i.\mathtt{init} = \mathtt{true}) \vee (\mathtt{M}_i.\mathsf{type} =$ "BRB"$))$ **do**
  $\vec{\mathtt{M}}_{id.\mathsf{src}} \xleftarrow{\cup} \mathtt{M}_i$
end for
if $\exists \vec{\mathtt{M}}' \in \vec{\Pi}[\vec{\mathtt{C}}_{\mathsf{PB}.id.\mathsf{src}}].\mathsf{TXT} : ((\vec{\mathtt{M}}'.\mathsf{tsid} = \vec{\mathtt{M}}.\mathsf{tsid}) \wedge (|\vec{\mathtt{M}}'| \neq |\vec{\mathtt{M}}_{id.\mathsf{src}}|))$ **then**
  $\mathtt{win} \leftarrow \mathtt{true}$
end if
return $\vec{\mathtt{M}}$

$\underline{\mathcal{O}\mathsf{Snd}(i, \vec{\mathtt{M}}, \vec{\mathtt{F}}, \vec{\mathtt{P}}) \rightarrow c}$

$st_i, \vec{\mathtt{C}} \leftarrow \mathsf{Ch.Snd}(st_i, \vec{\mathtt{M}}, \vec{\mathtt{F}}, \vec{\mathtt{P}})$
for $j \in |\vec{\mathtt{F}}|$ **do**
  if $(\vec{\mathtt{F}}_j.\mathsf{type} =$ "BIB"$\wedge (\vec{\mathtt{P}}_j \notin \mathsf{CORR})$ **then**
    for tar $\in \vec{\mathtt{F}}_j.\mathsf{targets}$ **do**
      // Keep track of Auth
      // queries ontag *and* msg
      // for SUF-CMA
      $\mathsf{AUTH} \xleftarrow{\cup} (\vec{\mathtt{C}}_{\mathrm{tar}}, \vec{\mathtt{C}}_j.\mathtt{tag}[\mathrm{tar}])$
    end for
  end if
  // Swap out $\vec{\mathtt{C}}_{\mathrm{tar}}$ w/rand for IND$
  // game
  if $(\vec{\mathtt{F}}_j.\mathsf{type} =$ "BCB"$\wedge (\vec{\mathtt{P}}_j \notin \mathsf{CORR}))$ **then**
    for tar $\in \vec{\mathtt{F}}_j.\mathsf{targets}$ **do**
      if $b = 1$ **then**
        $c \xleftarrow{\$} \{0,1\}^{|\vec{\mathtt{C}}_{\mathrm{tar}}|}$
        $\mathsf{CTXT}[c] \leftarrow (\vec{\mathtt{C}}_{\mathrm{tar}}, \vec{\mathtt{P}}_j)$
        $\vec{\mathtt{C}}_{\mathrm{tar}} \leftarrow c$
      end if
      if $b = 0$ **then**
        $\mathsf{CTXT}[\vec{\mathtt{C}}_{\mathrm{tar}}] \leftarrow (\vec{\mathtt{C}}_{\mathrm{tar}}, \vec{\mathtt{P}}_j)$
      end if
    end for
  end if
  $\vec{\Pi}[j].\mathsf{TXT} \xleftarrow{\cup} \vec{\mathtt{M}}$
end for
return $\vec{\mathtt{C}}$

$\underline{\mathsf{Corrupt}(i, \vec{\mathtt{P}})}$

$\mathsf{CORR} \xleftarrow{\cup} \vec{\mathtt{P}}$
return $st_i.k_{\vec{\mathtt{P}}}$

*Figure 4.14: Security Definition for Flexible Secure Channels (*FSC*). The modifications needed for* SFSC *experiment for the* StrongBPSec *with Read Receipts is presented in boxes. Refer to Table 4.2 for notational definitions.*

is negligible under the sufcma, aead, *and* dae *security of the* MAC, AEAD, DAE *primitives, respectively. Thus we have:* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{aead}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{A}}(\lambda)$.

Proof. We note that in the base FSC game there are three main ways the $\mathcal{A}$ can win. First, by forging a BIB tag, secondly by forging a BCB ciphertext, and thirdly by guessing the bit b. Thus we divide our proof into three cases: in the first case the adversary has caused $\texttt{win} \leftarrow \texttt{true}$ by generating a forged MAC tag, in the second case the adversary has caused $\texttt{win} \leftarrow \texttt{true}$ by forging an AEAD ciphertext, and finally in the third case the adversary guesses the challenger bit b but does not set $\texttt{win} \leftarrow \texttt{true}$. In each case we upper-bound the probability of the adversary causing the winning event to occur, and thus prove that the BPSec construction is FSC secure. We thus split our analysis into the following cases:

- C_1 : $\mathcal{A}$ sets $\texttt{win} \leftarrow \texttt{true}$ when a party π^i verifies $(\overrightarrow{\texttt{M}}'_t, \overrightarrow{\texttt{C}}_j.\texttt{tag}[t])$ such that $\overrightarrow{\texttt{M}}'_t \neq \perp$, but $(\overrightarrow{\texttt{M}}'_t, \overrightarrow{\texttt{C}}_j.\texttt{tag}[t]) \notin \mathsf{AUTH}$ for $\overrightarrow{\texttt{C}}_j.\mathsf{type} = \mathsf{BIB}$ and $\mathsf{Corrupt}(\cdot, \overrightarrow{\texttt{C}}_j.\overrightarrow{\texttt{P}})$ was not issued, i.e. $\mathcal{A}$ has successfully forged a valid message MAC tag pair for a BIB that verifies correctly and has guessed the challenge bit b.
- C_2: $\mathcal{A}$ sets $\texttt{win} \leftarrow \texttt{true}$ when a party π^i verifies $\overrightarrow{\texttt{C}}$ such that $\overrightarrow{\texttt{M}}_t \neq \perp$, $\overrightarrow{\texttt{C}}_t.\mathsf{type} = \mathsf{BCB}$, $p \leftarrow \mathsf{Extr}(\overrightarrow{\texttt{C}}_t, \mathsf{st}_i)$ and $\mathsf{Corrupt}(\cdot, \overrightarrow{\texttt{C}}_j, \overrightarrow{\texttt{P}})$ was not issued, i.e. $\mathcal{A}$ has successfully forged a valid AEAD ciphertext and has guessed the challenge bit b.
- C_3 $\mathcal{A}$ does not set $\texttt{win} \leftarrow \texttt{true}$, and has terminated the experiment and output a bit b', i.e. $\mathcal{A}$ has guessed the challenge bit b.

We say $\mathcal{A}$'s advantage breaking FSC security of BPSec is $\mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_i}(\lambda) \leq \mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_1}(\lambda) + \mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_2}(\lambda) + \mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_3}(\lambda)$.

We proceed via a sequence of games with each game focusing on a specific win condition (or lack thereof) in the three cases. We bound the difference in the adversary's advantage in each game with the underlying cryptographic assumptions until the adversary reaches a game where the advantage of that game equals 0, which shows that adversary $\mathcal{A}$ cannot win with non-negligible advantage.

Case 1 : π^i **accepts a forged message tag pair for a** BIB.

Game 0 This is the initial FSC security game. Thus $\mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 In this game, for any key-wrapping keys associated with parameter sets that have not been Corrupted, we abort if $\mathcal{A}$ forges a DAE ciphertext, and replace honestly

generated key-wrapped keys with uniformly random values. To do so, by a hybrid argument we introduce a series of reductions $\mathcal{B}_1$ as follows: At the beginning of the experiment, $\mathcal{B}_1$ will initialise a DAE challenger $\mathcal{C}_{\mathsf{DAE},i}$ for each parameter set $\overrightarrow{\mathtt{P}}_i$ such that $\mathsf{Corrupt}(\cdot, \overrightarrow{\mathtt{P}}_i)$ has not been issued (where i is the index into the hybrid argument). Whenever the challenger is required to key wrap ephemeral key k' using k_i, the challenger instead queries $(\emptyset, k')$ to the associated $\mathcal{C}_{\mathsf{DAE},i}$ and replaces the honestly generated key-wrap ciphertext with the output from $\mathcal{C}_{\mathsf{DAE},i}$. Additionally, each time $\mathcal{B}_1$ generates a ciphertext from $\mathcal{C}_{\mathsf{DAE},i}$, they maintain a table $\mathsf{DAE}[i] \xleftarrow{\cup} (C, k')$ which they update with each honestly generated key-wrap ciphertext. Whenever $\mathcal{B}_1$ is required to decrypt a DAE ciphertext using the key-wrap key associated with parameter set $\overrightarrow{\mathtt{P}}_i$, they first check if $(C, k') \in \mathsf{DAE}[i]$. If so, they use k' as the key-wrapped key. If not, they submit $(\emptyset, C)$ to $\mathcal{C}_{\mathsf{DAE},i}$ challenger.

Note that by definition of the DAE security game described in Definition 16, if $\mathcal{A}$ forges a DAE ciphertext and the bit b sampled by DAE is 0, then DAE will return the correct decryption of C, otherwise DAE returns $\perp$. Thus, any adversary that can forge a DAE ciphertext can be used to break the security of the DAE game. Additionally, we note that if the bit b sampled by $\mathcal{C}_{\mathsf{DAE},i}$ is 0, then the $\mathcal{C}_{\mathsf{DAE},i}$ encrypts the ephemeral key-wrapped keys k' honestly, and we are in **Game 0**. Otherwise, if the bit b sampled by the $\mathcal{C}_{\mathsf{DAE},i}$ is 1, then $\mathcal{C}_{\mathsf{DAE},i}$ simply samples a ciphertext uniformly at random, and now the key k' is uniformly random and completely independent of the protocol flow and we are in **Game 1**. Any $\mathcal{A}$ that could distinguish this change can be directly used to break the DAE security of the DAE primitive.

We note that $\mathcal{C}_{\mathsf{DAE},i}$ samples keys identically to the security experiment, and thus this replacement is sound. Finally, we note that $\mathcal{A}$ can never later call $\mathsf{Corrupt}(\cdot, \overrightarrow{\mathtt{P}}_i)$, so all internal values to DAE will never need to be revealed. As a result of these changes we know that any key-wrap output from a parameter set that has not been Corrupted is uniformly random and independent of the protocol flow, and $\mathcal{A}$ has no information on this key. There are at most n_a total parameter sets and thus n_a hybrid arguments and thus we find: $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" any of the MAC tags $\sigma_t \in \overrightarrow{\mathtt{C}}_j.\mathtt{tag}[t]$ for $t \in \overrightarrow{\mathtt{C}}_j.\mathsf{targets}$ of a block $j \in |\overrightarrow{\mathtt{C}}|$ such that $\overrightarrow{\mathtt{C}}_j.\mathsf{type} = \mathsf{BIB}$, $\overrightarrow{\mathtt{M}}_t \neq \perp$, and no honest party generated a message $\overrightarrow{\mathtt{M}}_t$. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge BIB using a key associated with a parameter set $p = \mathsf{Extr}(\overrightarrow{\mathtt{C}}_t, \mathsf{st}_i)$ such that $\mathsf{Corrupt}(\cdot, p)$ was not issued. We do so by introducing a series

of reductions $\mathcal{B}_2$ as follows: At the beginning of the experiment, for each parameter set $\overrightarrow{\mathtt{P}}_i$ that does not support key-wrapping $\mathcal{B}_2$ initialises a MAC challenger $\mathcal{C}_{\mathsf{sufcma},i}$. Additionally, during the experiment if a DAE ciphertext is generated or decrypted using k_{p_i} (where $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p_i)$), $\mathcal{B}_2$ initialises a MAC challenger $\mathcal{C}_{\mathsf{sufcma},i}$. Next, whenever $\mathcal{B}_2$ is required to generate a MAC tag over a message m using k_{p_i} (or the ephemeral key-wrapped key k' keywrapped under k_{p_i}) for p_i, $\mathcal{B}_2$ instead queries m to $\mathcal{C}_{\mathsf{sufcma},i}$. If $\overrightarrow{\mathtt{C}}_t$ verifies correctly but was not produced by an honest security source, then $(\overrightarrow{\mathtt{M}}_t, \sigma_t) \notin \mathsf{AUTH}$ and $\mathcal{A}$ has broken the sufcma security of the MAC. We note that by **Game 1** and the definition of the security experiment, all MAC keys replaced in this way were already uniformly random and independent of the protocol flow, and $\mathcal{A}$ cannot learn these by issuing a Corrupt query. We have at most n_a parameter sets that may be used directly as MAC keys, and at most n_m key-wrapped MAC keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

Case 1 Analysis: By definition of **Case 1** we know that $\mathcal{A}$ has caused $\mathtt{win} \leftarrow \mathtt{true}$ by forcing some party π^i to accept a message block $\overrightarrow{\mathtt{M}}_t$ such that $\overrightarrow{\mathtt{C}}_t.\mathsf{type} = \mathsf{BIB}$, $\mathsf{Corrupt}(\cdot, \mathsf{Extr}(\overrightarrow{\mathtt{C}}_t, st_i))$ has not been issued. By **Game 1** we replaced the ephemeral key k' used to generate the BIB if p is a key-wrapping algorithm. By **Game 2** we added an abort query that prevents $\mathcal{A}$ from forging MAC tags and thus BIB blocks. Since we abort if the adversary forges their own valid BIB, by **Game 2** π^i aborts and thus $\mathcal{A}$ cannot win the game. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_1}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$ We now proceed to **Case 2**.

Case 2 : π^i **accepts a forged message, ciphertext pair for a** BCB

Game 0 This is the initial FSC security game. Thus $\mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 This game proceeds identically to **Game 1** of **Case 1**, with the same reductions and bounds. Thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_3}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" a ciphertext block $\overrightarrow{\mathtt{C}}_t$ such that $\overrightarrow{\mathtt{C}}_t.\mathsf{type} = \mathsf{BCB}$, $\overrightarrow{\mathtt{C}}_t \neq \perp$, and no honest party generated a message $\overrightarrow{\mathtt{M}}_t$. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge BCB using a key associated with a parameter set $p = \mathsf{Extr}(\overrightarrow{\mathtt{C}}_t, \mathsf{st}_i)$ such that $\mathsf{Corrupt}(\cdot, p)$ was not issued. We do so by introducing a series of reductions $\mathcal{B}_4$ as follows: At the beginning of the experiment, for each parameter set $\overrightarrow{\mathtt{P}}_i$ that does not support key-wrapping, $\mathcal{B}_4$ initialises an AUTH-security AEAD challenger $\mathcal{C}_{\mathsf{AUTH},i}$. Additionally, during the experiment if a DAE ciphertext is generated or decrypted using k_{p_i} (where $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p_i)$), $\mathcal{B}_4$ initialises an AUTH AEAD challenger $\mathcal{C}_{\mathsf{AUTH},i}$.

Next, whenever $\mathcal{B}_4$ is required to generate a ciphertext over a message, nonce and header tuple (m, N, H) using k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for p_i, $\mathcal{B}_4$ instead queries (m, N, H) to $\mathcal{C}_{\mathsf{auth},i}$'s encryption oracle. Similarly, if $\mathcal{B}_4$ is required to decrypt a ciphertext, nonce, header tuple (C, N, H) for k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for parameter set p_i, $\mathcal{B}_4$ instead queries (C, N, H) to $\mathcal{C}_{\mathsf{AUTH},i}$'s decryption oracle. If $\overrightarrow{\mathtt{C}}_t$ decrypts correctly but C was not produced by an honest security source, then $(\overrightarrow{\mathtt{C}}_t, p_i) \notin \mathsf{CTXT}$ and $\mathcal{A}$ has broken the AUTH security of the AEAD scheme. Submitting (C, H, N) to $\mathcal{C}_{\mathsf{AUTH},i}$'s decryption oracle then allows $\mathcal{B}_4$ to win the AUTH-security game. We note that by **Game 1** and the definition of the security experiment, all AEAD keys replaced in this way were already uniformly random and independent of the protocol flow, and $\mathcal{A}$ cannot learn these by issuing a $\mathsf{Corrupt}$ query. We have at most n_a parameter sets that may be used directly as AEAD keys, and at most n_m key-wrapped AEAD keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{AUTH}}_{\mathcal{B}_4,\mathsf{AEAD}}(\lambda)$.

Case 2 Analysis: By definition of **Case 2** we know that $\mathcal{A}$ has caused $\mathtt{win} \leftarrow \mathtt{true}$ by forcing some party π^i to accept a message block $\overrightarrow{\mathtt{C}}_t$ such that $\overrightarrow{\mathtt{C}}_t.\mathsf{type} = \mathsf{BCB}$, and $\mathsf{Corrupt}(\cdot, \mathsf{Extr}(\overrightarrow{\mathtt{C}}_t, st_i))$ has not been issued. By **Game 1** we replaced the ephemeral key k' used to generate the BCB if p is a key-wrapping algorithm. By **Game 2** we added an abort query that prevents $\mathcal{A}$ from forging ciphertexts and thus BCB blocks. Since we abort if the adversary forges their own valid BCB, by **Game 2** π^i aborts and thus $\mathcal{A}$ cannot win the game. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_2}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{AUTH}}_{\mathcal{B}_2,\mathsf{AEAD}}(\lambda)$ We proceed to **Case 3**.

Case 3 : $\mathcal{A}$ has terminated the experiment and output a bit b', i.e. $\mathcal{A}$ has guessed the challenge bit b.

Game 0 This is the initial FSC security game. Thus $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A},C_3}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 This game proceeds identically to **Game 1** of **Case 1**, with the same reductions and bounds. Thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_5}(\lambda)$.

Game 2 In this game, all ciphertexts produced by un-$\mathsf{Corrupt}$ed party generating a ciphertext block $\overrightarrow{\mathtt{C}}_t$ such that $\overrightarrow{\mathtt{C}}_t.\mathsf{type} = \mathsf{BCB}$, using a key associated with a parameter set $p = \mathsf{Extr}(\overrightarrow{\mathtt{C}}_t, \mathsf{st}_i)$ such that $\mathsf{Corrupt}(\cdot, p)$ was not issued are replaced with uniformly random and independent strings. We do so by introducing a series of reductions $\mathcal{B}_6$ as follows: At the beginning of the experiment, for each parameter set $\overrightarrow{\mathtt{P}}_i$ that does not support key-wrapping $\mathcal{B}_6$ initialises an PRIV-security AEAD challenger $\mathcal{C}_{\mathsf{PRIV},i}$. Additionally, during the experiment if a DAE ciphertext is generated or decrypted using k_{p_i} (where $\mathcal{A}$ has not issued

$\mathsf{Corrupt}(\cdot, p_i)$), $\mathcal{B}_6$ initialises an PRIV AEAD challenger $\mathcal{C}_{\mathsf{PRIV},i}$.

Next, whenever $\mathcal{B}_6$ is required to generate a ciphertext over a message, nonce and header tuple (m, N, H) using k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for p_i, $\mathcal{B}_6$ instead queries (m, N, H) to $\mathcal{C}_{\mathsf{PRIV},i}$'s encryption oracle. Similarly, if $\mathcal{B}_6$ is required to decrypt a ciphertext, nonce, header tuple (C, N, H) for k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for parameter set p_i, $\mathcal{B}_6$ instead queries (C, N, H) to $\mathcal{C}_{\mathsf{PRIV},i}$'s decryption oracle. We note that by **Game 1** and the definition of the security experiment, all AEAD keys replaced in this way were already uniformly random and independent of the protocol flow, and $\mathcal{A}$ cannot learn these by issuing a Corrupt query.

If the bit b sampled by $\mathcal{C}_{\mathsf{PRIV},i}$ is 0, then the $\mathcal{C}_{\mathsf{PRIV},i}$ encrypts the message m honestly, and we are in **Game 1**. Otherwise, if the bit b sampled by the $\mathcal{C}_{\mathsf{PRIV},i}$ is 1, then $\mathcal{C}_{\mathsf{PRIV},i}$ simply samples a ciphertext uniformly at random, and now the ciphertext is uniformly random and completely independent of the protocol flow and we are in **Game 2**. Any $\mathcal{A}$ that could distinguish this change can be directly used to break the PRIV security of the PRIV primitive. We have at most n_a parameter sets that may be used directly as AEAD keys, and at most n_m key-wrapped AEAD keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{priv}}_{\mathcal{B}_2,\mathsf{AEAD}}(\lambda)$.

Case 3 Analysis: By **Game 1** we replaced the ephemeral key k' used to generate the BCB if p is a key-wrapping algorithm. By **Game 2** we replaced all ciphertexts with uniformly random strings regardless of the challenge bit b. Since the behaviour of the security experiment no longer relies of the challenge bit b the $\mathcal{A}$ has no advantage in guessing and thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{FSC},\mathcal{A}}_{\mathsf{BPSec},C_3}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_5}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{PRIV}}_{\mathcal{B}_6,\mathsf{AEAD}}(\lambda)$.

□

We now turn to analysing the SFSC security of the StrongBPSec construction.

Theorem 7 (SFSC Security for StrongBPSec)**.** *The* StrongBPSec *protocol presented in Figures 4.11, 4.12 and 4.13 is* SFSC*-secure. That is, for any PPT algorithm $\mathcal{A}$ against the* SFSC *security experiment (described in Definition 37)* $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec}}(\lambda)$ *is negligible under the* sufcma *and* dae *security of the* MAC *and* DAE *primitives respectively. Thus we have:* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{BPSec},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{A}}(\lambda)$.

Proof. We note that in the strong FSC game there are five main ways $\mathcal{A}$ can win. In addition to those described in Theorem 6, the SFSC game introduces two new ways for $\mathcal{A}$ to set `win` $\leftarrow$ `true`. In particular, if $\mathcal{A}$ has successfully dropped message blocks sent by the

security source without being detected when the bundle is processed by the security target, or by forging a read receipt for a given message block. The analysis for the three original cases from the FSC game apply here, and so we focus on the other two cases:

C_1: Adversary $\mathcal{A}$ has successfully dropped blocks $\overrightarrow{\mathtt{M}}_i$ from the source-constructed bundle $\overrightarrow{\mathtt{C}}$ without detection and the bundle payload integrity block $\mathsf{BIB_B}$ verifies correctly;

C_2: Adversary$\mathcal{A}$ has successfully forged a read receipt BRB that verifies correctly.

We proceed via a sequence of games, and bound the difference in the $\mathcal{A}$'s advantage in each game with the underlying cryptographic assumptions. We conclude when the adversary reaches a game where the advantage of that game for that case equals 0, which shows that $\mathcal{A}$ cannot win with non-negligible advantage. We begin by dividing the proof into two separate cases corresponding to each win condition (and denote with $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_i}(\lambda)$ the advantage of the adversary in winning the strong integrity game in Case i). It follows that $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_i}(\lambda) \leq \mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_1}(\lambda) + \ldots + \mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_5}(\lambda)$.
We define with $\mathsf{Adv}^{\mathcal{A}}_{G_i}(\lambda)$ the advantage of $\mathcal{A}$ in Game i. We begin with Case 1.

Case 1: Adversary $\mathcal{A}$ wins by causing some party π^i to accept bundle $\overrightarrow{\mathtt{C}}$ from a security source $\pi^{\overrightarrow{\mathtt{C}}_{\mathtt{PB}.id.\mathtt{src}}}$ with missing blocks $\overrightarrow{\mathtt{M}}_i$. By the definition of the case, we know that the adversary $\mathcal{A}$ has not been able to $\mathsf{Corrupt}$ any keys $(k_p, k_p^{\mathtt{RR}}, k_p^{\mathtt{BB}})$ such that $k_p^{\mathtt{BB}}$ was used to verify (or key-wrap keys to verify) the $\mathsf{BIB_B}$. We proceed via the following series of games.

Game 0 This is the SFSC security game in Case 1: $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 This game proceeds identically to **Game 1** of **Case 1** of Theorem 6, with the same reductions and bounds. Thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-$\mathsf{Corrupt}$ed party "accepts" a message $\overrightarrow{\mathtt{M}}$ and no honest party generated a message with the same payload and of the same length. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge $\mathsf{BIB_B}$ using a key associated with a parameter set $\overrightarrow{\mathtt{P}}_i$ such that $\mathsf{Corrupt}(\cdot, \overrightarrow{\mathtt{P}}_i)$ was not issued. This game proceeds identically to **Game 2** of **Case 1** of **Theorem 1**, with the same reductions and bounds. We have at most n_a parameter sets that may be used directly as MAC keys, and at most n_m key-wrapped MAC keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

Case 1 Analysis: By definition of **Case 1** we know that $\mathcal{A}$ has caused $\mathtt{win} \leftarrow \mathtt{true}$ by forcing some party π^i to accept a message with "missing blocks" $\overrightarrow{\mathtt{M}}$. Specifically, if π^i accepts a message whilst incorrectly believing that the original bundle had a different

amount of blocks than generated. By definition of the security experiment we know that $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p)$ such that π^i used $k_p^{\mathtt{BB}}$ to verify the BIB_B block. By **Game 1** we replaced the ephemeral key k' used to generate the BIB_B if p is a key-wrapping algorithm. By **Game 2** we added an abort query that prevents $\mathcal{A}$ from forging BIB_B blocks. We note that the BIB_B is computed over each key identifier that was used by the source node SN and the number of target blocks that the key identifier was used for. Thus, if π^i believed that SN generated an incorrect number of blocks, then the bundle integrity block BIB_B for $\overrightarrow{\mathtt{M}_{\mathtt{PLD}}}$ would not verify correctly. Since we abort if the adversary forges their own valid BIB_B, by **Game 2** π^i aborts and thus $\mathcal{A}$ cannot win the game. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_1}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$. Now we transition to **Case 2**.

Case 2: $\mathcal{A}$ wins by causing a party π^i to accept a read receipt BRB under key $k_p^{\mathtt{RR}}$ but no read receipt was generated by an honest party.. By the definition of the case, we know that the adversary $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p)$. We proceed via the following series of games.

Game 0 This is the SFSC security game in Case 2: $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 Let n_a be the total number of parameter sets used in the experiment. In this game, for any key-wrapping keys associated with parameter sets that have not been Corrupted, we abort if $\mathcal{A}$ forges a DAE ciphertext, and replace honestly generated key-wrapped keys with uniformly random values. This proceeds identically to the reduction described in **Game 1** of **Case 1** with the same factors thus we find: $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" a message $\overrightarrow{\mathtt{M}}$ with a read receipt BRB (generated using $k_p^{\mathtt{RR}}$) and no honest party generated BRB. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge BRB using a key associated with a parameter set $\overrightarrow{\mathtt{P}}_i$ such that $\mathsf{Corrupt}(\cdot, \overrightarrow{\mathtt{P}}_i)$ was not issued. This game proceeds similarly (up to a change in notation) to **Game 2** of **Case 1** of **Theorem 1**, with the same reductions and bounds. We have at most n_a parameter sets that may be used directly as MAC keys, and at most n_m key-wrapped MAC keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

Case 2 Analysis: By **Case 2** we know that a party π^i that verifies a BRB using $k_p^{\mathtt{RR}}$ (or a key-wrapped key encrypted under $k_p^{\mathtt{RR}}$) that sets $\mathtt{win} \leftarrow \mathtt{true}$ such that $\mathcal{A}$ has not issued $\mathsf{Corrupt}(i, p)$ will abort. Thus $\mathcal{A}$ cannot win the game in **Case 2** and thus we

have $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{SFSC},\mathcal{A}}_{\mathsf{StrBPSec},C_2}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

□

4.7 Conclusion

In this chapter we presented the first formalization and cryptographic security analysis of deep space communication protocol BPSec. We introduced a flexible channel formalism FSC in order to model and capture the peculiarities of BPSec compared to channel protocols on the terrestrial internet. We analyzed the security of BPSec for security goals it claims to offer and highlighted its weakness against an arbitrary message dropping adversary. We proposed StrongBPSec as stronger variant of BPSec that provides additional integrity guarantees against arbitrary message dropping between a bundle source and its intended final destination.

Part IV

Design of Secure Avionic Protocols

Chapter 5

Key Exchanges & Hybrid Authenticated Key Exchanges

In subsequent chapters we venture to propose novel protocol constructions for secure avionic communication. Specifically, we propose novel key establish constructions PQAG for aviation in Chapter 6 and a secure avionic handover construction PQAG-HO in Chapter 7. In order to formalize our constructions and cryptographically analyze their security, we dedicate this chapter to present our formalization for key exchanges in Section 5.2, followed by the modified HAKE security framework in Section 5.3 (originally proposed by Dowling et al.[50]) which integrates our KEX formalization. The modified HAKE model presented in Section 5.3 will be used for the analysis of our PQAG and PQAG-HO constructions presented in Sections 6.2 and 7.2 respectively. We begin this chapter with a brief discussion on the current state-of-the-art in post-quantum cryptography and its broader implications on the field of cryptography as a whole.

5.1 Classical & Post-Quantum Key Exchange

Diffie and Hellman [48] in their seminal work introduced the concept of securely establishing a shared key over an insecure public channel, that only required sharing public components of the corresponding key material for the key establishment. Bellare and Rogaway [19] provided the first formalization of security for such key exchange schemes, capturing the security properties of entity authentication and session key secrecy. Their model assumes an adversary that fully controls the communication between interacting parties; the adversary

is allowed to read, modify, delay, replay or inject messages as they wish. Later work further extended the Bellare–Rogaway model to capture various other security properties; forward secrecy [49]; compromise of ephemeral secrets [39, 78]; and three-party key exchanges [20]. However, it must be noted that these key exchange schemes and their security assume a *classical adversary limited by their computational power* to break the underlying hard mathematical problems in polynomial time.

In fact, Shor [113] has proposed an algorithm that is capable of solving the problem of factorizing large integers as well as discrete logarithms— the two central mathematical problems on which most current cryptosystems are built— in polynomial time provided that the algorithm is carried out by a quantum computer with a large number of quantum bits [43]. Consequently, the security of prevalent cryptographic applications— such as key exchange schemes based on the hardness of solving discrete logarithms— will be challenged when the advent of the quantum computer is fully realized. The situation is further exacerbated by the threat of an attacker who captures and archives sensitive network traffic, intending to decrypt it later with the advent of quantum computers. Within this context, there is a reinvigorated interest— among academic and industry researchers alike— to strengthen the security of existing communication infrastructure with novel cryptographic primitives that are resistant to threats posed by a quantum-enabled attacker. Indeed, reinforcing the urgency of the matter, the National Institute of Standards and Technology (NIST) has initiated a competition-like process [7] with the objective of standardizing appropriate post-quantum primitives. At the time of writing, after three rounds, NIST has already chosen some winners for standardization in key encapsulation (KYBER) and digital signatures (DILITHIUM, FALCON, SPHINCS+) categories. NIST has further declared an additional fourth round with the intention of selecting more contenders for standardization as post-quantum key encapsulation primitives. Table 5.1 summarizes the five major families of existing PQC approaches and their underlying mathematical problems along with examples of known primitives for each category.

However, it must also be noted that the security of many existing post-quantum schemes is still relatively immature, with less rigorous analysis compared to classical counterparts. Recent attacks on the digital signature scheme Rainbow [24] and the KEM SIKE [41], both previously NIST PQC finalists, highlight the evolving nature of these post-quantum primitives. Within this context, hybrid approaches that combine classical and post-quantum primitives offers a pragmatic solution, providing protection against both potential flaws in emerging PQC algorithms and future quantum threats.

PQ Family	Mathematical Problem
Lattice-based	Based on lattice problems like Shortest Vector Problem (SVP), Closest Vector Problem (CVP), Shortest Independent Vectors Problem (SIVP). E.g., CRYSTALS-KYBER, CRYSTALS-DILITHIUM, FALCON
Hash-based	Collision resistance of a hash function. E.g., SPHINCS+
Isogeny-based	Finding the isogeny mapping between two elliptic curves with the same number of points. E.g., SIDH (insecure), CSIDH.
Code-based	Hardness of decoding in a random linear code. E.g., BIKE, Classic McEliece (both NIST 4^{th} round contenders).
Multivariate	Solving a set of multivariate quadratic equations that are proven to be NP-hard. E.g., Rainbow (insecure)

Table 5.1: Post-quantum cryptographic families.

Many academic and industry researchers alike have already been studying novel ways to construct public-key cryptosystems resistant to attacks carried out by a future-quantum or quantum adversary. For instance, at the point of writing, all websites and APIs served through Cloudflare support post-quantum hybrid key agreement by combining the classical X25519 Elliptic-curve Diffie-Hellman and the new post-quantum Kyber512 and Kyber768 for TLS 1.3 handshake [129]. Moreover, Amazon Web Services have integrated quantum-resistant hybrid (Diffie-Hellman+Kyber) security across their core services that leverage their TLS implementations s2n-tls and s2n-quic, including the AWS Transfer Family that uses SSH for SFTP file transfers [38, 67].

The hybrid authenticated key exchange (HAKE) framework proposed by Dowling et al. [50] introduces a flexible formalization that is capable of analyzing the security of such hybrid authenticated key exchange (AKE) schemes that combine multiple classical and post-quantum primitives. The HAKE framework extends the classic Bellare and Rogaway [19] model to incorporation additional security notions of *perfect forward secrecy* and *post-compromise security* while accommodating for the strengths of both classical and post-quantum adversaries. The work of Bindel et al. [25] also proposes a framework that capture

the security of hybrid authenticated key exchange protocols but their work is exclusively limited to hybrid KEM schemes. In comparison, the HAKE framework offers flexibility by supporting KEM constructions while remaining adaptable to other cryptographic primitives, making it suitable for analyzing the security of a wide range of hybrid AKE schemes.

The avionic key establishment construction, PQAG, and the aviation handover scheme, PQAG-HO, introduced in Chapters 6 and 7, respectively, integrate cryptographic components designed to ensure security against potential quantum adversaries. We harness the modified HAKE framework we discuss in this chapter for the security analysis of our proposed constructions. We further discuss other constructions modeled within the original HAKE framework that are either foundational (e.g. , Muckle [50]) or concurrent and independent to our work (Muckle+ [34] and Muckle# [16]) in Section 6.2.4. In this chapter, we simplify the HAKE framework for the analysis of our proposed protocols in Chapters 6 and 7. Moreover, we expound on the concept of "probabilistic key-exchange protocol Π" HAKE is built around, replacing it with our own simple formalization for a key exchange scheme KEX.

5.2 Secure Key Exchange Formalization

In this section we formalise our notion of secure key exchange KEX protocols, explaining the expected functionality, phases and outputs. We utilize this framework in the construction of all our secure avionic communication schemes presented in Chapters 6 and 7.

5.2.1 KEX Syntax

A key exchange protocol KEX, illustrated in Figure 5.1, consists of a tuple of algorithms $\mathsf{KEX} = (\mathsf{KGen}, \mathsf{SetUp}, \mathsf{Proc})$. We use π_i^s to refer to both the identifier of the s-th instance of the KEX being run by party P_i and the collection of per-session variables maintained for the s-th instance of KEX run by P_i. We describe the algorithms below:

– $\mathsf{KGen}(\lambda) \xrightarrow{\$} (pk, sk, \mathsf{pid})$ is a probabilistic key generation algorithm taking as input a security parameter λ and outputting a public-key/secret-key pair (pk, sk) and (potentially) a party identifier pid. During the protocol execution KGen is independently run by each party engaging in the KEX scheme. This algorithm models an long-term key-pair generation by individual participants.

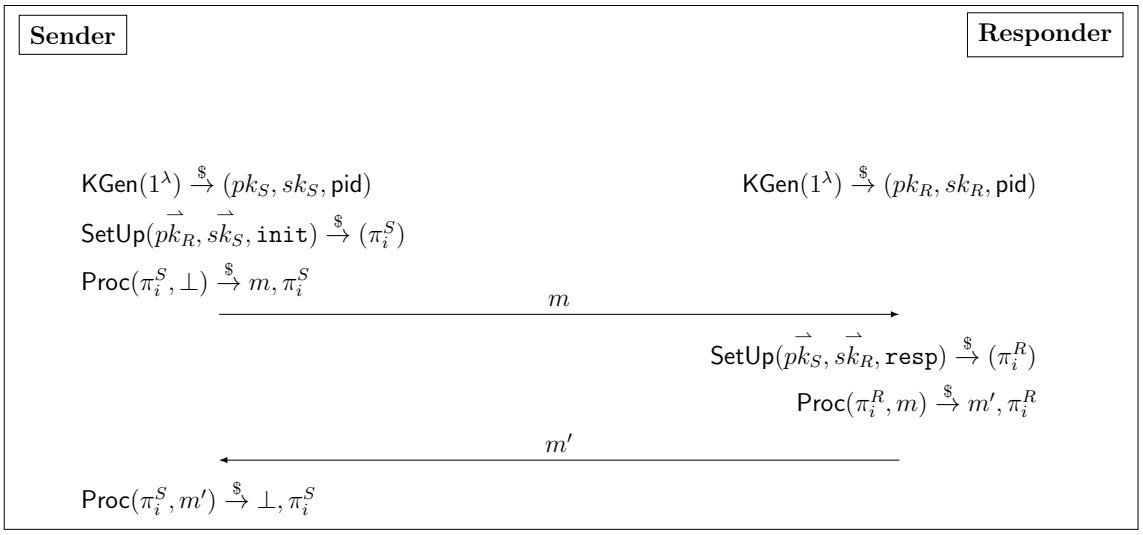


Figure 5.1: Generic KEX construction.

- $\mathsf{SetUp}(\overrightarrow{pk}_j, \overrightarrow{sk}_i, \rho) \xrightarrow{\$} (\pi_i^s)$ is a probabilistic algorithm that takes as input the public-key output of the corresponding party's KGen execution $\overrightarrow{pk}_j$ (potentially a set of public keys), the secret key $\overrightarrow{sk}_i$ (potentially a set of secret keys) from its own KGen execution and their role (denoted by ρ) in the KEX session, either `init` or `resp`, and outputs some state π_i^s. This algorithm is only run once per session by each party. SetUp models the computation of shared keys by participants during an execution of KEX.
- $\mathsf{Proc}(\pi, m) \xrightarrow{\$} (m_i, \pi)$ is a probabilistic algorithm that takes as input some state π generated by SetUp and, if it exists, message $m \in \{0,1\}^* \cup \{\emptyset\}$ produced by a previous execution of Proc, and outputs some message $m' \in \{0,1\}^* \cup \{\emptyset\}$ and an updated per-session state π'. This algorithm models final session key agreement and session update by participants.

In the following section we integrate our KEX formalism within the HAKE framework.

5.3 The HAKE Model

As in the original model [50], the modified HAKE model leveraged for our analysis is based on Bellare-Rogaway-based AKE models, and captures adversaries of differing strength (quan-

tum and classical) via a detailed key compromise interface. Specifically, we model quantum adversaries by allowing them to compromise non-post-quantum key establishment primitives (for instance, elliptic curve-based algorithms to establish a shared secret key). The majority of our modifications from the original HAKE model are simplifications – since HAKE explicitly captures the use of preshared symmetric keys, not used within our new PQAG protocols. We give a high-level description of the modified HAKE framework in Section 5.3.2 (and detail the differences between our variant and the original HAKE framework). We then describe cleanness and partnering definitions in Section 5.3.4 as well as Section 5.3.5.

5.3.1 Secret Key Generation

Recall that HAKE addresses secret key generation (the output of a "KGen" algorithm) of individual key establishment primitives explicitly, and categorises them into *long-term* (i.e. generated once and used in every execution of the protocol), and *ephemeral* (i.e. generated on a per-stage basis) secret generation. We simplify the HAKE model by only including the following sub-categories:

- *Post-quantum asymmetric secret generation* – long-term post-quantum asymmetric secrets (for example, signature secret keys), are generated by LQKGen, whereas ephemeral post-quantum asymmetric secrets (such as KEM secret keys) are generated by EQKeyGen.
- *Classical asymmetric secrets* – long-term classical asymmetric secrets (for example, EdDSA secret keys) are generated by LCKGen, whereas ephemeral classical asymmetric secrets (for example, ECDH secret keys) are generated by ECKeyGen.

In addition, our proposed PQAG protocols are not a multi-stage key exchange protocols, which establishes multiple keys throughout protocol execution. Thus, we remove all multi-stage specific state and indexing in the HAKE execution environment. With this context, we now formally define the HAKE execution environment, capturing how an adversary can interact with a hybrid AKE protocol.

5.3.2 Execution Environment

Consider an experiment $\mathsf{Exp}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(\lambda)$ played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. $\mathcal{C}$ maintains a set of n_P parties $P_1, \ldots, P_{n_P}$ (representing users interacting with each

other in protocol executions), each capable of running up to (potentially parallel) n_S sessions of a probabilistic key exchange protocol KEX. Each session is an execution of the key exchange protocol KEX, represented as a tuple of algorithms $\mathsf{KEX} = (\mathsf{KGen}, \mathsf{SetUp}, \mathsf{Proc})$. $\mathsf{KEX.KGen}$ outputs a set of long-term keys $(\mathsf{LQKGen}, \mathsf{LCKGen})$ and $\mathsf{KEX.SetUp}$ outputs a set of ephemeral keys $(\mathsf{EQKeyGen}, \mathsf{ECKeyGen})$ for the HAKE experiment. We use π_i^s to refer to both the identifier of the s-th instance of the KEX being run by party P_i and the collection of per-session variables maintained for the s-th instance of KEX run by P_i, and $\mathsf{KEX.Proc}$ is an algorithm capturing the honest execution of the protocol KEX by protocol participants. We describe generically these algorithms below:

(a) $\mathsf{KEX.KGen}(\lambda) \xrightarrow{\$} (pk, sk)$ is a set of $\mathsf{KEX.LXKGen}(\lambda)$ algorithms, where $X \in \{\mathsf{C}, \mathsf{Q}\}$. $\mathsf{KEX.LXKGen}$ is a probabilistic *post-quantum long-term* (if $\mathsf{X} = \mathsf{Q}$), or a *classic long-term* (if $\mathsf{X} = \mathsf{C}$) asymmetric key generation algorithm, taking a security parameter λ and outputting a public-key/secret-key pair (pk, sk).

(b) $\mathsf{KEX.SetUp}(\overrightarrow{pk_j}, \overrightarrow{sk_i}) \xrightarrow{\$} (\pi)$ is a set of $\mathsf{KEX.EYSetUp}(\overrightarrow{pk_j}, \overrightarrow{sk_i}) \xrightarrow{\$} \pi$ algorithms, where $Y \in \{\mathsf{C}, \mathsf{Q}\}$. $\mathsf{KEX.EYSetUp}$ embeds all relevant ephemeral keys for the current session π_i^s within its output state. $\mathsf{KEX.EYSetUp}$ is a probabilistic *post-quantum ephemeral* (if $\mathsf{Y} = \mathsf{Q}$), or *classic ephemeral* (if $\mathsf{Y} = \mathsf{C}$) asymmetric key generation algorithm, taking as input a vector of long-term key pairs $(\overrightarrow{pk_j}, \overrightarrow{sk_i})$ and outputting a state π which embeds ephemeral keys $(\overrightarrow{pk}, \overrightarrow{sk})$ for the current session π_i^s.

(c) $\mathsf{KEX.Proc}(\pi, m) \xrightarrow{\$} (m', \pi')$ is a (potentially) probabilistic algorithm that takes as input some state π generated by $\mathsf{KEX.SetUp}$ and, if it exists, message $m \in \{0,1\}^* \cup \{\emptyset\}$ produced by a previous execution of $\mathsf{KEX.Proc}$, and outputs some message $m' \in \{0,1\}^* \cup \{\emptyset\}$ and an updated per-session state π'. The input state π for $\mathsf{KEX.Proc}$ (potentially) embeds relevant security parameters λ, the set of long-term asymmetric key pairs $\vec{pk}_i, \vec{sk}_i$ of the party P_i, a collection of per-session variables π and an arbitrary bit string $m \in \{0,1\}^* \cup \{\emptyset\}$. $\mathsf{KEX.Proc}$ outputs a response $m' \in \{0,1\}^* \cup \{\emptyset\}$ and an updated per-session state π', behaving as an honest protocol implementation.

$\mathcal{C}$ runs $\mathsf{KEX.LQKGen}(\lambda)$ and $\mathsf{KEX.LCKGen}(\lambda)$ n_P times to generate long-term post-quantum and long-term classical asymmetric key pairs (which we denote with $\vec{pk}_i, \vec{sk}_i$) for each party $P_i \in \{P_1, \ldots, P_{n_P}\}$, and delivers all public-keys $\vec{pk}_i$ for $i \in \{1, \ldots, n_P\}$ to $\mathcal{A}$. The challenger $\mathcal{C}$ then randomly samples a bit $b \xleftarrow{\$} \{0,1\}$ and interacts with $\mathcal{A}$ via the

queries listed in Section 5.3.3, also maintaining a set of corruption registers, representing a list of ephemeral and long-term secrets that have been compromised by $\mathcal{A}$ via Reveal, Corrupt and Compromise queries. Eventually, $\mathcal{A}$ issues a Test query, to which $\mathcal{C}$ responds with k_b, either the real session key generated by the Test session (when $b = 0$), or a random key from the same distribution (when $b = 1$). $\mathcal{C}$ now interacts with $\mathcal{A}$ via the queries listed in Section 5.3.3 (except the Test query), and eventually terminates and outputs a guess d of the challenger bit b. The adversary $\mathcal{A}$ wins the HAKE key-indistinguishability experiment if $d = b$, and additionally if the test session π satisfies a cleanness predicate clean, which we discuss in more detail in Section 5.3.5. We give an algorithmic description of this experiment in Figure 5.2.

$\mathsf{Exp}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(\lambda)$:

$b \xleftarrow{\$} \{0,1\}$
$\mathsf{KEX.LXKGen}(\lambda) \xrightarrow{\$} pqpk_i, pqsk_i, cpk_i, csk_i \ \forall i \in [n_P]$
$\vec{pk}_i \leftarrow (pqpk_i, cpk_i)$
$\mathsf{LQK}_i, \mathsf{LCK}_i \leftarrow \mathtt{clean} \ \forall i \in [n_P], j \in [n_P]$
$\mathsf{EQ}\vec{\mathsf{K}}^s_i, \mathsf{EC}\vec{\mathsf{K}}^s_i \leftarrow \mathtt{clean} \ \forall i \in [n_P], s \in [n_S]$
$ctr \leftarrow 0$
$d \xleftarrow{\$} \mathcal{A}(\vec{pk})^{\mathsf{Send},\mathsf{Create}\star,\mathsf{Corrupt}\star,\mathsf{Compromise}\star,\mathsf{Reveal}}$
if $\mathsf{clean}(\pi_b)$ **then return** $(d = b)$
elsereturn $d \xleftarrow{\$} \{0,1\}$
end if

$\mathsf{Send}(i, s, m)$:

if $\pi^s_i.\alpha \neq \mathtt{active}$ **then return** $\perp$
end if
$\mathsf{KEX.Proc}(\pi^s_i, m) \xrightarrow{\$} (\pi^s_i, m')$
if $\pi^s_i.\alpha = \mathtt{reject}$ **then return** $\perp$
end if
$\pi^s_i.\mathbf{m}_r \leftarrow \pi^s_i.\mathbf{m}_r \| m$
$\pi^s_i.\mathbf{m}_s \leftarrow \pi^s_i.\mathbf{m}_s \| m'$ **return** m'

$\mathsf{Reveal}(i, s)$:

if $\pi^s_i.\alpha \neq \mathtt{accept}$ **then return** $\perp$
end if
$\mathsf{S}\vec{\mathsf{K}}^s_i \leftarrow \mathtt{corrupt}$
if $\exists (j,t)$ s.t. $\mathsf{match}(\pi^s_i, \pi^t_j)$ **then**
  $\mathsf{S}\vec{\mathsf{K}}^t_j \leftarrow \mathtt{corrupt}$
end ifreturn $\pi^s_i.\mathbf{k}$

$\mathsf{Test}(i, s)$:

if $\pi_b \neq \perp$ **then return** $\perp$
end if
if $\pi^s_i.\alpha \neq \mathtt{accept}$ **then return** $\perp$
end if
$k_0 \xleftarrow{\$} \mathcal{D}$, $k_1 \leftarrow \pi^s_i.\mathbf{k}$
$\mathsf{S}\vec{\mathsf{K}}^s_i \leftarrow \mathsf{tested}$
$\pi_b \leftarrow \pi^s_i$ **return** k_b

$\mathsf{Create}(i, j, role)$:

let $s = min\{s : \pi^s_i.\rho = \perp\}$
$\pi^s_i.\rho = role$
$\pi^s_i.pid = j$
$\mathsf{KEX.EYSetUp}(\vec{pk}_j, \vec{sk}_i) \rightarrow \pi^s_i$
$\pi^s_i.\mathbf{eqk} \leftarrow \pi^s_i.pqsk, \pi^s_i.\mathbf{eck} \leftarrow \pi^s_i.pqsk$
return s

$\mathsf{CorruptCK}(i)$:

if $\mathsf{LCK}_i = \mathtt{corrupt}$ **then return** $\perp$
end if
$\mathsf{LCK}_i \leftarrow \mathtt{corrupt}$ **return** csk_i

$\mathsf{CorruptQK}(i)$:

if $\mathsf{LQK}_i = \mathtt{corrupt}$ **then return** $\perp$
end if
$\mathsf{LQK}_i \leftarrow \mathtt{corrupt}$ **return** $pqsk_i$

$\mathsf{CompromiseQK}(i, s)$:

if $\mathsf{EQ}\vec{\mathsf{K}}^s_i = \mathtt{corrupt}$ **then return** $\perp$
end if
$\mathsf{EQ}\vec{\mathsf{K}}^s_i \leftarrow \mathtt{corrupt}$ **return** $\pi^s_i.\mathbf{eqk}$

$\mathsf{CompromiseCK}(i, s)$:

if $\mathsf{EC}\vec{\mathsf{K}}^s_i = \mathtt{corrupt}$ **then return** $\perp$
end if
$\mathsf{EC}\vec{\mathsf{K}}^s_i \leftarrow \mathtt{corrupt}$ **return** $\pi^s_i.\mathbf{eck}$

Figure 5.2: HAKE *experiment for an adversary* $\mathcal{A}$ *against the key-indistinguishability security of protocol* KEX. *Note that the values* $\vec{pk}$ *given as input to* $\mathcal{A}$ *and* $\mathcal{Q}$ *represent the vector* $\vec{pk}_i$ *for all* n_P *parties. The function* match *takes as input two sessions* π^s_i *and* π^t_j *and determines if they are* matching *according to some matching definition. For the definition of the matching sessions function used in our* HAKE *experiment, see Section 5.3.4.*

Each session maintains a set of per-session variables:

– $\rho \in \{\mathtt{init}, \mathtt{resp}\}$: The role of the party in the current session. Note that parties can

be directed to act as `init` or `resp` in concurrent or subsequent sessions.
- $pid \in \{1, \ldots, n_P, \star\}$: The intended communication partner, represented with $\star$ if unspecified. Note that the identity of the partner session may be set during the protocol execution, in which case *pid* can be updated once.
- $\alpha \in \{\texttt{active}, \texttt{accept}, \texttt{reject}, \perp\}$: The status of the session, initialised with $\perp$.
- $\mathbf{m}_i \in \{0,1\}^* \cup \{\perp\}$, *where* $i \in \{\mathbf{s}, \mathbf{r}\}$: The concatenation of messages sent (if $i = \mathbf{s}$) or received (if $i = \mathbf{r}$) by the session at each stage, initialised by $\perp$.
- $\mathbf{k} \in \{0,1\}^* \cup \{\perp\}$: The session key, or $\perp$ if no session key has yet been computed.
- $\mathbf{exk} \in \{0,1\}^* \cup \{\perp\}$, where $\mathbf{x} \in \{\mathbf{q}, \mathbf{c}\}$: The *post-quantum ephemeral asymmetric* (if $\mathbf{x} = \mathbf{q}$), or *classic ephemeral asymmetric* (if $\mathbf{x} = \mathbf{c}$) secret key generated by the session, initialised by $\perp$.
- $\mathbf{st} \in \{0,1\}^*$: Any additional state used by the session at each stage.

5.3.3 Adversarial Interaction

Our HAKE framework considers a traditional AKE adversary, in complete control of the communication network, able to modify, inject, delete or delay messages. They are able to compromise several layers of secrets: (a) long-term private keys, allowing our model to capture forward-secrecy notions and quantum adversaries. (b) ephemeral private keys, modelling the leakage of secrets due to the use of bad randomness generators, or potentially bad cryptographic primitives or quantum adversaries. (c) session keys, modelling the leakage of keys by their use in bad cryptographic algorithms. The adversary interacts with the challenger $\mathcal{C}$ via the queries below:

- $\mathsf{Create}(i, j, role) \rightarrow \{(s), \perp\}$: Allows the adversary $\mathcal{A}$ to initialise a new session owned by party P_i, where the role of the new session is ρ, and intended communication partner party P_j. If a session π_i^s has already been created, $\mathcal{C}$ returns $\perp$. Otherwise, $\mathcal{C}$ returns (s) to $\mathcal{A}$.
- $\mathsf{Send}(i, s, m) \rightarrow \{m', \perp\}$: Allows $\mathcal{A}$ to send messages to sessions for protocol execution and receive the output. If the session $\pi_i^s.\alpha \neq \texttt{active}$, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ computes $\mathsf{KEX.Proc}(\pi_i^s, m) \rightarrow (m', \pi_i^{s\prime})$, sets $\pi_i^s \leftarrow \pi_i^{s\prime}$, updates transcripts $\pi_i^s.\mathbf{m}_r$, $\pi_i^s.\mathbf{m}_s$ and returns m' to classical/post-quantum adversaries $\mathcal{A}$.
- $\mathsf{Reveal}(i, s)$: Allows $\mathcal{A}$ access to the session keys computed by a session. $\mathcal{C}$ checks if $\pi_i^s.\alpha = \texttt{accept}$ and if so, returns $\pi_i^s.\mathbf{k}$ to $\mathcal{A}$. In addition, the challenger checks if there exists another session π_j^t that *matches* with π_i^s, and also sets $\vec{\mathsf{SK}}_j^r \leftarrow \texttt{corrupt}$.

Otherwise, $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.

- $\mathsf{Test}(i, s) \rightarrow \{k_b, \perp\}$: Allows $\mathcal{A}$ access to a real-or-random session key k_b used in determining the success of $\mathcal{A}$ in the key-indistinguishability game. If a session π_i^s exists such that $\pi_i^s.\alpha = \texttt{accept}$, then the challenger $\mathcal{C}$ samples a key $k_0 \xleftarrow{\$} \mathcal{D}$ where $\mathcal{D}$ is the distribution of the session key, and sets $k_1 \leftarrow \pi_i^s.\mathbf{k}$. $\mathcal{C}$ then returns k_b (where b is the random bit sampled during set-up) to $\mathcal{A}$. Otherwise $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.
- $\mathsf{CorruptXK}(\{i, j\}) \rightarrow \{k_i, \perp\}$: Allows $\mathcal{A}$ access to the secret post-quantum long-term key $pqsk_i$ (if $\mathsf{X} = \mathsf{Q}$) or the secret classical long-term key csk_i (if $\mathsf{X} = \mathsf{C}$), generated for the party P_i (and P_j, in the preshared case) prior to protocol execution. If the secret long-term key has already been corrupted previously, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.
- $\mathsf{CompromiseYK}(i, s) \rightarrow \{\mathbf{eqk}, \mathbf{eck}, \perp\}$: Allows $\mathcal{A}$ access to the secret ephemeral post-quantum key $\pi_i^s.\mathbf{eqk}$ (if $\mathsf{Y} = \mathsf{Q}$), or the secret ephemeral classical key $\pi_i^s.\mathbf{eck}$ (if $\mathsf{Y} = \mathsf{C}$) generated for the session π_i^s prior to protocol execution. If $\pi_i^s.\mathbf{eqk}/\pi_i^s.\mathbf{eck}$ has already been corrupted previously, then $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$.

5.3.4 Partnering Definition

To determine which secrets $\mathcal{A}$ can reveal without trivially breaking the security of a given session, our model must define how sessions are *partnered*. In our work, we use the notion of *matching sessions* [77], and *origin sessions* [47] to construct our security analysis. On a high level, π_i^s is an origin session of π_j^t if π_i^s has received the messages that π_j^t sent without modification, even if the reply that π_i^s sent back has not been received by π_j^t. If all messages sent and received by π_i^s and π_j^t are identical, then the sessions *match*. We give detailed definitions and a precise pseudocode description of these functions as follows:

Definition 39 (Matching Sessions)**.** *We consider $\pi_i^s.\mathbf{m}_s$ and $\pi_i^s.\mathbf{m}_r$ to be the concatenation of all messages sent and received (respectively) by a session π_i^s. We say that π_i^s* matches *a session π_j^t in stage t if $\pi_i^s.pid = j$, $\pi_j^t.pid = i$, $\pi_i^s.\rho \neq \pi_j^t.\rho$, $\pi_i^s.\mathbf{m}_r = \pi_j^t.\mathbf{m}_s$ and $\pi_i^s.\mathbf{m}_s = \pi_j^t.\mathbf{m}_r$.*

We now define origin sessions for use in the HAKE security experiment.

Definition 40 (Origin Sessions)**.** *We consider $\pi_i^s.\mathbf{m}_s$ and $\pi_i^s.\mathbf{m}_r$ to be the concatenation of all messages sent and received (respectively) by a session π_i^s. We say that π_i^s* matches *a session π_j^t if $\pi_j^t.\mathbf{m}_s = \pi_i^s.\mathbf{m}_r$. We say that π_i^s* prefix-matches *a session π_j^t if $\pi_j^t.\mathbf{m}_s = \pi_i^s.\mathbf{m}'_r$ where $\pi_i^s.\mathbf{m}'_r$ is $\pi_i^s.\mathbf{m}_s$ truncated to the length of $|\pi_j^t.\mathbf{m}_s|$. Finally, we say that a session π_i^s*

has an origin session *with* π_j^t *if* π_i^s *prefix-matches* π_j^t *(and* π_i^s *has sent the last message) or* π_i^s *matches* π_j^t *(and* π_j^t *has sent the last message).*

If all messages sent and received by π_i^s and π_j^t are identical, then the sessions *match*. We give precise pseudocode description of these functions in Figure 5.3.

$\mathsf{match}(\pi_i^s, \pi_j^t) \rightarrow \{0,1\}$:

1: **if** $(\pi_i^s.\mathbf{m}_s \neq \pi_j^t.\mathbf{m}_r)$
2: $\quad \vee(\pi_i^s.\mathbf{m}_r \neq \pi_j^t.\mathbf{m}_s)$
3: $\quad \vee(\pi_i^s.\rho = \pi_j^t.\rho) \vee(\pi_i^s.pid \neq j)$
4: $\quad \vee(\pi_j^t.pid \neq i)$ **then return** 0
5: **end ifreturn** 1

$\mathsf{origin}(\pi_i^s, \pi_j^t) \rightarrow \{0,1\}$:

1: **if** $(\pi_i^s.\mathbf{m}_r \neq \pi_j^t.\mathbf{m}_s) \vee$
2: $(\pi_i^s.\mathbf{m}'_r \neq \pi_j^t.\mathbf{m}_s : \pi_i^s.\mathbf{m}'_r$
3: $\quad = \quad \mathsf{trunc}(\pi_i^s.\mathbf{m}_r, |\pi_j^t.\mathbf{m}_s|)$
then return 0
4: **end ifreturn** 1

Figure 5.3: A pseudocode description of the matching session *and* origin session *functions.*

5.3.5 Cleanness Predicates

Cleanness predicates in authenticated key exchange protocols detail the exact restrictions on adversarial powers. For instance, in protocols that are not post-compromise secure, the leakage of the long-term key of a party trivially allows the adversary to impersonate that party. Thus, it follows that sessions established after that corruption (with that party as the communicating peer) cannot be secure. We note that the cleanness predicates defined below are specific to our PQAG-KEM and PQAG-SIG constructions discussed in Chapter 6.

Briefly, the PQAG protocols defend against a quantum adversary. Thus, a successful adversary is allowed to compromise the long-term and ephemeral classical asymmetric secrets (via CorruptCK and CompromiseCK, respectively) without penalty. Since the PQAG protocols authenticate with post-quantum primitives, and aim to achieve perfect forward secrecy, we allow a successful adversary to issue a $\mathsf{CorruptQK}(j)$ query (where the $\pi_i^s.pid = j$ and $\mathsf{Test}(i,s)$ was queried), as long as π_i^s was completed before the $\mathsf{CorruptQK}(j)$ query was issued.

Thus, a "clean" session has not had $\mathcal{A}$ compromise: (a) the ephemeral post-quantum secrets of the Test session and its matching partner in the tested stage, (b) the long-term post-quantum secrets of the Test session's partner before the Test session completes. We formalise this intuition as $\mathsf{clean}_{q\mathsf{HAKE}}$ in Definition 41.

Definition 41 ($\mathsf{clean}_{q\mathsf{HAKE}}$)**.** *A session π_i^s such that $\pi_i^s.\alpha = \texttt{accept}$ and $\pi_i^s.pid = j$ in the security experiment defined in Figure 5.2 is $\mathsf{clean}_{q\mathsf{HAKE}}$ if all of the following conditions hold:*

1. *The query $\mathsf{Reveal}(i, s)$ has not been issued.*
2. *For all $(j, t) \in n_P \times n_S$ such that π_i^s matches π_j^t, the query $\mathsf{Reveal}(j, t)$ has not been issued.*
3. *If there exists a session π_j^t such that π_j^t matches π_i^s, then the following sets of queries has not been issued:*
   - $\mathsf{CompromiseQK}(i, s)$, $\mathsf{CompromiseQK}(j, t)$ *have not been issued, where π_j^t* matches *π_i^s.*
4. *If there exists no $(j, t) \in n_P \times n_S$ such that π_j^t is an origin session of π_i^s, then $\mathsf{CorruptQK}(j)$ has not been issued before $\pi_i^s.\alpha \leftarrow \texttt{accept}$.*

It may also be desirable to determine the security guarantees that PQAG provides against classical adversaries in the event of a new vulnerability discovered in the underlying post-quantum key establishment primitive, as demonstrated by the recent attacks against Rainbow [24], or SIDH [42, 82, 103].

Thus, a "clean" session has not had $\mathcal{A}$ compromise: (a) *either* the ephemeral classic secrets of the Test session and its matching partner in the tested stage, *or* (b) the ephemeral post-quantum secrets of the Test session and its matching partner in the tested stage, *and* (c) the long-term post-quantum secrets of the Test session's partner before the Test session completes. In order to capture this scenario, we formalise this intuition as $\mathsf{clean}_{c\mathsf{HAKE}}$ in Definition 42.

Definition 42 ($\mathsf{clean}_{c\mathsf{HAKE}}$)**.** *A session π_i^s such that $\pi_i^s.\alpha = \texttt{accept}$ and $\pi_i^s.pid = j$ in the security experiment defined in Figure 5.2 is $\mathsf{clean}_{q\mathsf{HAKE}}$ if all of the following conditions hold:*

1. *The query $\mathsf{Reveal}(i, s)$ has not been issued.*

2. *For all* $(j,t) \in n_P \times n_S$ *such that* π_i^s matches π_j^t, *the query* $\mathsf{Reveal}(j,t)$ *has not been issued.*
3. *If there exists a session* π_j^t *such that* π_j^t *matches* π_i^s, *then at least one of the following sets of queries has not been issued:*
   - $\mathsf{CompromiseQK}(i,s)$, $\mathsf{CompromiseQK}(j,t)$ *have not been issued, where* π_j^t matches π_i^s.
   - $\mathsf{CompromiseCK}(i,s)$, $\mathsf{CompromiseCK}(j,t)$ *have not been issued, where* π_j^t matches π_i^s.
4. *If there exists no* $(j,t) \in n_P \times n_S$ *such that* π_j^t *is an origin session of* π_i^s, *then* $\mathsf{CorruptCK}(j)$ *has not been issued before* $\pi_i^s.\alpha \leftarrow \texttt{accept}$.

Finally, we formalise the advantage of an adversary $\mathcal{A}$ in winning the HAKE key indistinguishability experiment in the following way:

Definition 43 (HAKE Key Indistinguishability)**.** *Let* KEX *be a key-exchange protocol, and* n_P, $n_S \in \mathbb{N}$. *For a particular given predicate* clean, *a QPT algorithm* $\mathcal{A}$, *we define the advantage of* $\mathcal{A}$ *in the* HAKE *key-indistinguishability game to be* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(()\lambda) = 2 \cdot \left|\Pr\left[\mathsf{Exp}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(()\lambda) = 1\right] - \frac{1}{2}\right|$.

We say that KEX *is post-quantum* HAKE*-secure if, for all QPT algorithms* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(()\lambda)$ *is negligible in the security parameter* λ. *We say that* KEX *is classically* HAKE*-secure if, for all PPT algorithms* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(()\lambda)$ *is negligible in the security parameter* λ.

Chapter 6

Post Quantum Key Exchange Protocols for Aviation

In this chapter we propose two hybrid key exchange protocols (PQAG-SIG and PQAG-KEM) to secure CPDLC communication between ground stations (G) and aircrafts (A). We provide a formal proof of security for the proposed protocols against both quantum and classical adversaries. We instantiate our proposed protocols with different post-quantum algorithms to understand the real-world applicability of our protocols to the resource-constrained ecosystem of avionic communication. We compare our proposals to existing work and demonstrate that we provide satisfactory performance with the added benefit of heightened (hybrid) security. We begin this chapter with a discussion on the current state of secure communication within the aviation sector.

6.1 Secure Communication in Aviation

Controller-Pilot Data Link Communications (CPDLC) is a protocol that facilitates communication between the Air Traffic Control (ATC) stations and aircrafts over a datalink medium. CPDLC was designed to reduce communication loads on the Very High Frequency (VHF) band, in order to handle large numbers of aircrafts communicating with ATCs simultaneously in congested air traffic zones. CPDLC communication has multiple applications, ranging from route change and clearances to level assignments and crossing constraints [71]. At present all CPDLC communications are carried out over unencrypted and unauthenti-

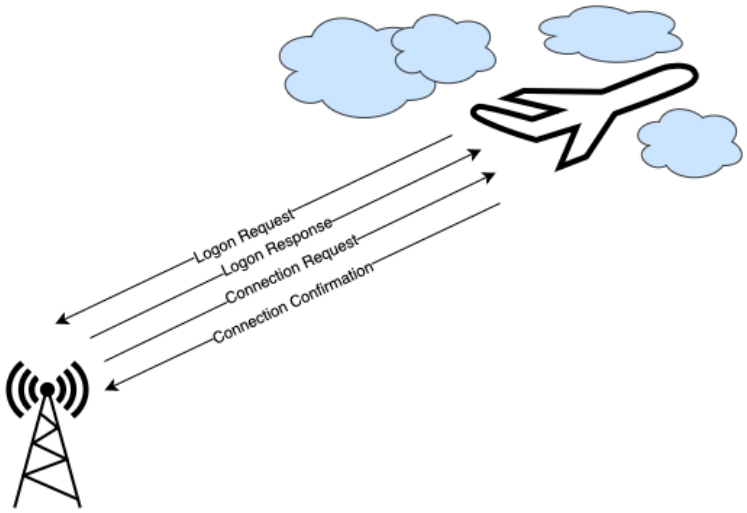


Figure 6.1: A generic insecure CPDLC communication between ground station G *and aircraft* A.

cated datalinks. Figure 6.1 illustrates an expected implementation of CPDLC as described by official ICAO guidelines [65]. CPDLC begins with a logon stage, where the aircraft A initiates a connection with the ground station G, which uniquely identifies A to G along with departure and destination locations for its journey. G responds to a logon request by informing A whether the logon request has been successful or not. Upon successful logon, G initiates a CPDLC connection by sending a connection request which A will accept, provided no existing CPDLC connection is active. After establishment, data link messages are exchanged between A and G that include clearances, instructions, and reports. The communication concludes with disconnection, either when the flight transitions to a different ground station's airspace or when CPDLC services are no longer required. We emphasize that this entire exchange occurs entirely in plaintext, leaving it exposed to potential interception and manipulation.

Therefore, there is a critical need to develop a future-proof secure communication protocol that both ensures confidentiality and authenticity of communication, while withstanding the evolving threat landscape. Considering the aviation industry's reluctance toward frequent upgrades due to technological challenges, safety considerations, and cost concerns, these protocols must ensure robust security against both current classical adversaries and future quantum attackers.

At the time of writing, NIST's Post Quantum Cryptography (PQC) Standardization Process has already standardized a selection of post-quantum candidates in key encapsulation and digital signatures categories. The objective of this process was to select post-quantum public key cryptographic primitives to potentially replace existing quantum-vulnerable primitives currently in widespread use. In addition, the White House recently published a memorandum [123], introducing a plan to increase resources and efforts for the multi-year process of migrating vulnerable computer systems to quantum-resistant cryptography.

However, almost all PQC increases computation and/or key sizes compared to quantum-vulnerable counterparts, complicating their practical adoption, especially in resource-constrained environments. For instance, the novel LDACS media for G-to-A communication provides a throughput of 303.33 kbps (forward-link) and 199.73 kbps (reverse-link) [22], and the average throughput of data-links used in avionic communication is 31.5kbps across the globe. Moreover, security analysis of PQC is relatively immature and constantly evolving compared to their classical counterparts [50]— corroborated by recent attacks undermining the security of digital signature scheme Rainbow [24] and the KEM SIKE [41]— both candidates previously shortlisted as NIST PQC finalists. Accordingly, adopting a hybrid approach [50] combining classical and PQC primitives presents a realistic equilibrium, providing security guarantees against potentially undiscovered vulnerabilities (both algorithmic and in-code) in novel PQC and security against quantum adversaries.

Next, we discuss existing work related to classical and post-quantum avionic communication protocols as well as other constructions (e.g. KEMTLS) that is comparable to our work.

6.1.1 Design Considerations for Secure Avionic Communication

The Automatic Dependent Surveillance-Broadcast (ADS-B) is a mandatory broadcast system for all aircrafts, where an aircraft periodically broadcasts its position, allowing it to be tracked. Wesson et. al. [128] discuss the security of the ADS-B protocol and highlight a set of considerations for aviation communication design: interoperability with existing policies and laws, bandwidth and interference constraints and how ADS-B operates in a *cryptographically untrusted environment*. Their work primarily evaluates symmetric and asymmetric settings, concluding that maintaining the secrecy of symmetric keys across multiple untrusted global domains is unsuitable for the purpose of secure communication

in aviation. The use of shared keys also complicates key revocation scenarios in the event of key compromise, since it requires replacing keys across all parties involved.

6.1.2 Symmetric Constructions

The novel ADS-B construction proposed by Yang et al. [130] is based on symmetric-key primitives; format-preserving encryption for obscuring flight identity and TESLA protocol [98] for authentication. Although computationally less expensive, their work heavily relies on a trusted third party for key management and distribution on a frequent basis. Furthermore, due its inherent design of delayed key disclosure, the use of TESLA protocol introduces additional latency to the scheme. Briefly, delayed key disclosure in the TESLA protocol ensures message authenticity by revealing the cryptographic key used for signing messages only after a predefined delay, preventing forgery during transmission. Implementing this in constrained environments can be challenging due to the need for synchronized clocks between sender and receiver and the added computational and storage overhead for buffering messages until the corresponding keys are disclosed. Our protocols (presented in Section 6) avoid these issues, as they are constructed from asymmetric-key primitives, require only a single round for key establishment, and secure CPDLC communications specifically, not addressing ADS-B.

6.1.3 Asymmetric Constructions

Other approaches to securing communication in aviation rely on asymmetric encryption schemes, introducing communication overhead compared to symmetric encryption, due to large public-key and ciphertext sizes. For instance, to achieve the symmetric-key equivalent strength of 112 bits (ADS-B packet size), which NIST claims is cryptographically secure until 2030, the ECDSA signature length is 448 bits, four times greater than an ADS-B message [128].

An identity-based solution for authenticating ADS-B has been proposed by the work of [124] that uses batch verification, which verify a *batch* of message and signature pairs in a single instance, for increased efficiency. Aside from the difficulty of maintaining the reliance of a trusted authority across multiple geographic regions for key management, the signature size of their proposed scheme is 512-bits added to the original 112-bit ADS-B packet size. The work of Asari et al. [9] proposes an anonymous authentication protocol for ADS-B based on Certificateless Public Key Cryptography (CL-PKC). However, their work involves

expensive bilinear pairings for signature generation and does not analyze how the inclusion of signatures affect the ADS-B packet sizes. The ideal-lattice based key exchange scheme of Yang et al. [132] somewhat superficially resembles our proposed protocol. However, unlike their work, our proposed schemes incorporate hybrid security from both classical and post-quantum cryptographic domains and prove their security against a post-quantum adversary in the HAKE model whereas [132] only proves their security against a polynomial time attacker in the eCK-PFS model [47].

Khan et. al. [71] propose a lightweight protocol for securing CPDLC using elliptic-curve cryptography and Schnorr signature schemes. Their method provides authenticated communication while preserving confidentiality and non-repudiation properties against a classical Dolev-Yao adversary, but fails to achieve security against a quantum adversary. Mäurer et. al. [86] also propose a secure alternative to CPDLC, which uses different variations of Diffie-Hellman key exchange (DHKE) to establish keys, and evaluate the practicality of their proposals. They modify the Station-to-Station (STS) protocol with distinct DHKE instantiations, comparing Elliptic Curve DHKE (ECDH) and Supersingular Isogeny DHKE (SIDH) as the underlying key exchange. Their work concludes that while STS-ECDH provides the most resource efficient performance, STS-SIDH (believed at the time to provide post-quantum security) was the better option for long-term security. We implement our PQAG construction for aviation in Section 6.3, and compare with Mäurer et. al.'s STS-(C)SIDH protocol [86], demonstrating our protocol's relative practicality while also achieving stronger notions of security. Moreover, the work of Mäurer et al. [86] does not verify their constructions with any formal security analysis.

The works of Bellido-Mangenell et. al. [22] and Mielke et. al. [90] also propose a secure protocol for CPDLC, and simulate their protocol's communication between aircrafts and ground stations. Their proposal combines the asymmetric post-quantum McEliece encryption scheme and symmetric encryption. The authors [90] highlight that the protocol does not guarantee security against potential man-in-the-middle attackers since the exchanged (ephemeral) public keys are not authenticated. Moreover, neither work provide any formal proof of security. Finally, Tiepelt et al. [125] formally analyze the security of proposed LDACS mutual authentication and key agreement protocol MAKE [88]. However MAKE is a *quantum-ready* construction and allows key negotiation through either a classical Diffie-Hellman key exchange or a post-quantum key encapsulation mechanism (KEM). In comparison, our PQAG protocols presented in Chapter 6 provides hybrid security guarantees against both classical and post-quantum adversaries. Furthermore, in contrast to [88],

we instantiate and implement our constructions to demonstrate the practical applicability of our proposed schemes.

6.1.4 Our Contributions

We propose two hybrid key exchange protocols to secure CPDLC communication between ground stations (G) and aircrafts (A). We provide a formal proof of security for the proposed protocols against both quantum and classical adversaries. We instantiate our proposed protocols with different post-quantum algorithms to understand the real-world applicability of our protocols to the resource-constrained ecosystem of avionic communication. Instantiating both approaches allowed us to evaluate their relative benefits and costs. We can see that there exists a trade-off between the two approaches: using signatures as the basis for authentication is more computationally expensive, but less costly in bandwidth. Moreover, placing both approaches in the context of packet sizes further demonstrates their relative practicality in the aviation domain. We compare our proposals to existing work and demonstrate that we provide satisfactory performance with the added benefit of heightened (hybrid) security.

6.2 PQAG Key Exchange Protocols

In this section we describe two variants of our proposed post-quantum air-to-ground communications protocol PQAG, PQAG-KEM and PQAG-SIG, executed between an Aircraft A and a Ground Station G. We give the detailed cryptographic operations in Figures 6.2 and 6.3. Note that PQAG-KEM and PQAG-SIG have slightly different infrastructure requirements, as well as different communication and computational overheads. In PQAG-SIG, we require no predistribution of public keys, but imposes higher communication costs due to large public-key sizes. Overheads added by PQAG-SIG are particularly high if the implementation utilizes post-quantum digital signatures (see Tables 6.4 and 6.5). In PQAG-KEM, we assume that A already knows G's public keys, reducing communication and computational overhead, which is a realistic (and scalable) assumption in avionics, where travel paths (and thus, G partners) can be known ahead of time.

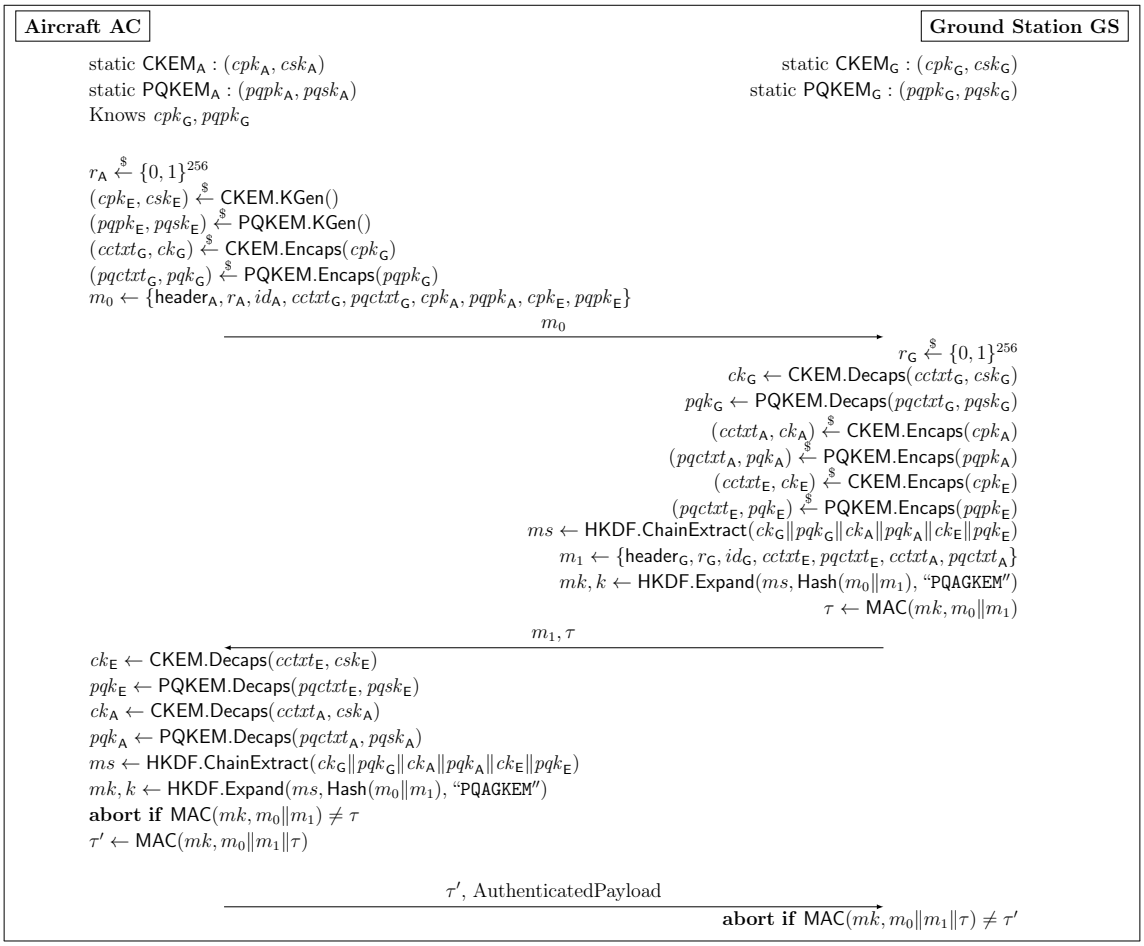


Figure 6.2: The PQAG-KEM *key exchange protocol. Note that* $\mathsf{HKDF.ChainExtract}\ (a \| b \| \ldots \| n) = \mathsf{HKDF.Extract}(\ldots \mathsf{HKDF.Extract}(\mathsf{HKDF.Extract}(a, 0), b) \ldots, n)$.

6.2.1 PQAG-KEM

Broadly, PQAG-KEM executes a series of post-quantum and classical KEMs between A and G, combining the outputs into symmetric keys mk and k. A and G maintain two long-term KEM key pairs, one classical and one post-quantum. We assume that G has pre-distributed their public KEM key pairs to A prior to the protocol execution, and that public keys can be validated by some PKI, outside the scope of our protocol. The MAC tags, computed using mk (itself computed using outputs of the long-term KEMs), provide mutual authentication, and the ephemeral KEMs provide forward secrecy for the derived session key k. We describe the protocol execution as follows:

- A begins by generating post-quantum and classical KEM ephemeral public keys ($pqpk_{\mathsf{E}}, cpk_{\mathsf{E}}$ respectively) and a random nonce r_A. Next, A encapsulates secrets under the long-term key pairs of G (cpk_{G} and $pqpk_{\mathsf{G}}$), computing ciphertexts $cctxt_{\mathsf{G}}$ and $pqctxt_{\mathsf{G}}$. Afterwards, A forms a message m_0 by concatenating some (arbitrary) header information $\mathsf{header}_{\mathsf{A}}$, with r_{A}, id_{A}, $cctxt_{\mathsf{G}}$, $pqctxt_{\mathsf{G}}$,$cpk_{\mathsf{E}}, pqpk_{\mathsf{E}}$ and the long-term public-keys of A $cpk_{\mathsf{A}}, pqpk_{\mathsf{A}}$, sending m_0 to G.

- Upon receiving m_0, G decapsulates post-quantum and classical ciphertexts $pqctxt_{\mathsf{G}}$ and $cctxt_{\mathsf{G}}$, deriving keys pqk_{G} and ck_{G}. Next, G encapsulates secrets under the ephemeral public keys $cpk_{\mathsf{E}}, pqpk_{\mathsf{E}}$, generating $cctxt_{\mathsf{E}}$ and $pqctxt_{\mathsf{E}}$ respectively. G further encapsulates secrets under A's long-term public keys $cpk_{\mathsf{A}}, pqpk_{\mathsf{A}}$, generating $cctxt_{\mathsf{A}}$ and $pqctxt_{\mathsf{A}}$. The key outputs of KEM encapsulations and decapsulations are used to derive a master key $ms = \mathsf{HKDF.ChainExtract}(ck_{\mathsf{G}} \| pqk_{\mathsf{G}} \| ck_{\mathsf{A}} \| pqk_{\mathsf{A}} \| ck_{\mathsf{E}} \| pqk_{\mathsf{E}})$. We note that HKDF.ChainExtract() consolidates a consecutive series of key derivations using the associated keying material. G forms message m_1 by concatenating some (arbitrary) header $\mathsf{header}_{\mathsf{G}}$, with r_{G}, id_{G}, $cctxt_{\mathsf{E}}$, $pqctxt_{\mathsf{E}}$, $cctxt_{\mathsf{A}}$ with $pqctxt_{\mathsf{A}}$. G derives a final session key k and MAC key mk by computing $mk, k = \mathsf{HKDF.Expand}(ms, \mathsf{Hash}(m_0 \| m_1), \text{“}\texttt{PQAGKEM}''\text{)}$. The G further computes $\tau \leftarrow \mathsf{MAC}(mk, m_0 \| m_1)$, sending m_1 and τ to A.

- A decapsulates $cctxt_{\mathsf{E}}$, $pqctxt_{\mathsf{E}}$, $cctxt_{\mathsf{A}}$ and $pqctxt_{\mathsf{A}}$ and derives the shared keys mk, k. Next, A verifies τ under mk and aborts the session if it fails. Finally, A computes $\tau' \leftarrow \mathsf{MAC}(mk, m_0 \| m_1 \| \tau)$ and returns τ' to G for authentication, and can immediately use k to establish secure communications with G. G finally verifies τ', aborting the session

if it fails.

6.2.2 PQAG-SIG

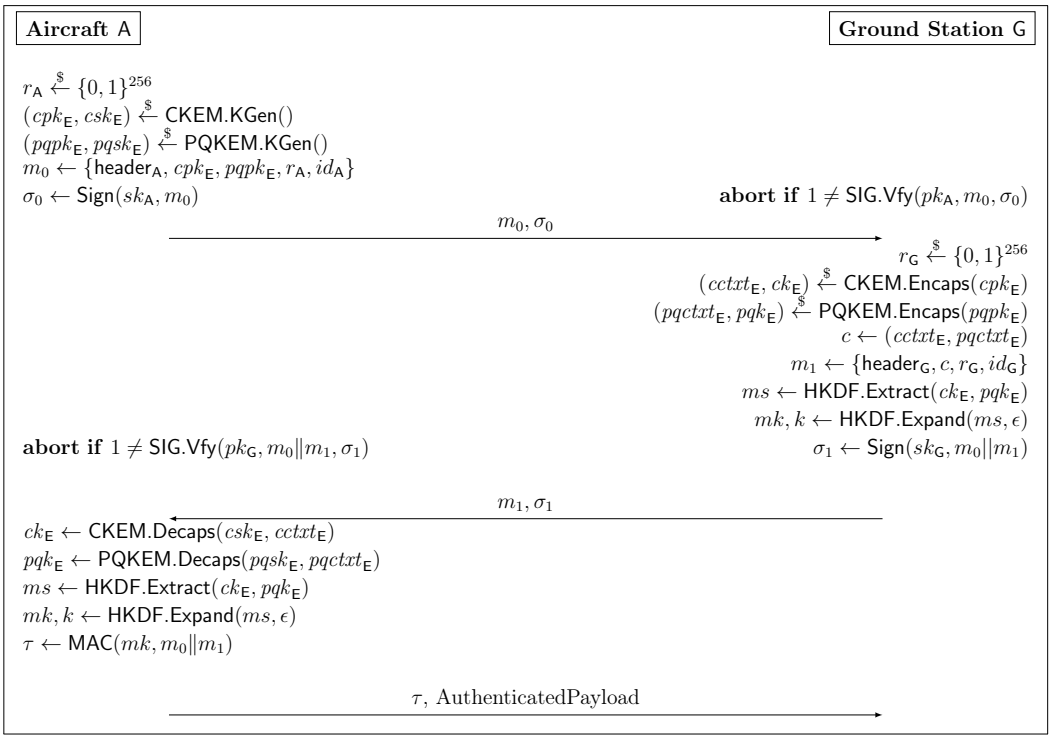


Figure 6.3: The PQAG-SIG *key exchange protocol.*

PQAG-SIG proceeds similarly to PQAG-KEM by combining ephemeral post-quantum and classical KEM outputs into a single session key k, achieving forward secrecy. However, authentication is achieved by post-quantum signature schemes SIG. Unlike PQAG-KEM, we do not assume any pre-distribution of public keys, but similarly assume that long-term public keys can be validated through some PKI outside the scope of our protocol. We describe the protocol execution as follows:

– A begins by generating post-quantum and classical KEM public keys $cpk_{\mathsf{E}}, pqpk_{\mathsf{E}}$ respectively and a random nonce r_{A}. A straightforwardly computes a message m_0 by concatenating header information $\mathsf{header}_{\mathsf{A}}$, cpk_{E}, $pqpk_{\mathsf{E}}$, r_{A} and its unique identifier id_{A}. A signs m_0 using its long-term secret key sk_{A} (outputting σ_0), sending m_0 and σ_0 to G.

– G verifies σ_0, aborting the session if it fails. G then encapsulates post-quantum and classical secrets under $pqpk_{\mathsf{E}}$, cpk_{E}, outputting $pqctxt_{\mathsf{E}}$ and $cctxt_{\mathsf{E}}$ respectively, which are concatenated into c. G forms message m_1 by concatenating header_B, ciphertext c, random nonce r_{G} and its unique identifier id_{G}. G derives an intermediate value $ms = \mathsf{HKDF.Extract}(ck, pqk)$, and derives the session key k and MAC key mk by computing $mk, k = \mathsf{HKDF.Expand}(ms, \epsilon)$. G generates its own signature $\sigma_1 \leftarrow \mathsf{Sign}(sk_{\mathsf{G}}, m_0 \| m_1)$ sending σ_1 and m_1 to A.

– A verifies σ_1 using G's long-term public key sk_{G}, and upon failure will abort the session. A then decapsulates $cctxt_{\mathsf{E}}$ and $pqctxt_{\mathsf{E}}$, and derives the shared keys mk, k. Finally, A computes $\tau \leftarrow \mathsf{MAC}(mk, m_0 \| m_1)$ and returns τ to G, immediately using k to communicate securely with G.

6.2.3 KEMTLS versus PQAG

KEMTLS [111] is a key exchange protocol, proposed to transition TLS 1.3 handshakes to a post-quantum setting. KEMTLS proposes the use of KEMs instead of digital signatures for server authentication, as post-quantum signature public keys and signatures tend to be larger than their post-quantum KEM counterparts. While TLS 1.3 commonly runs in unilateral (server-only) authentication, there are many scenarios which require mutual client-server authentication, which includes our air-to-ground communication setting. Schwabe et al. [112] propose a mutually-authenticated variant of KEMTLS, called KEMTLS-PDK, relying on pre-distributed server public keys.

While TLS 1.3's setting and requirements are similar to our own, the avionic communication media places significant restraints on both bandwidth and computation, as it abhors congestion and communication must complete within a restricted timeframe, necessitating the introduction of a custom protocol. Our two proposed protocols introduced in Section 6, PQAG-SIG and PQAG-KEM, superficially resemble TLS 1.3 and KEMTLS-PDK respectively, but our design has been specifically tailored to the domain constraints of avionic communication. Our key schedule is significantly simplified, avoiding the complexity introduced in TLS 1.3. In addition, we achieve 1.5 round-trip times for both variants. The PQAG-SIG protocol, introduced to take advantage of existing certificate-based infrastructures, mutually authenticates aircrafts and ground-control stations using certificates (classical or post-quantum) and derives a post-quantum hybrid shared key. Concurrently, the PQAG-KEM variant of our protocol provides post-quantum hybrid implicit mutual au-

thentication and derives a post-quantum hybrid shared key. Additionally, the payload sizes per RTT for all our PQAG-SIG and PQAG-KEM variants are significantly smaller compared to the existing KEMTLS instantiations [111]. For instance, using Kyber as the underlying cryptographic parameter the full cost of transmission for mutually authenticated KEMTLS, Cached TLS 1.3 and KEMTLS-PDK are respectively 9554, 10140 and 6324 bytes [111]. In comparison, all our instantiations of both PQAG-SIG and PQAG-KEM have significantly reduced transmission cost, particularly our hybrid instantiations, with PQAG-KEM-Hy requiring only 4584bytes per transmission (Table 6.5).

Moreover, KEM (and signature) combiners have been studied previously in literature, the focus has been on providing a mechanism for key-indistinguishability (or authentication), and not in their use in a higher-level cryptographic protocol. While one can view our work as a specific KEM combiner, we build AKEs with properties beyond IND-CCA (CPA) security, like forward secrecy and authentication. While previous works showed that if one of the underlying KEMs is secure, then the construction is too, our attacker can also expose the other KEM secrets, and thus is not implied directly by KEM combiners.

The work that approaches ours most is by Bindel et al. [25] who builds a generic compiler for hybrid AKEs from KEM combiners, but is a 2.5RTT protocol inspired by KEMTLS, which is unsuitable for similar reasons (high TLS-specific overhead, complex key schedule, etc.).

6.2.4 Muckle versus PQAG

The Muckle family of protocols constructed within the HAKE model, (Muckle [50], Muckle+ [34] and Muckle# [16]), while not designed specifically for aviation, are either foundational or concurrent and independent to our work. As such, they provide a basis for superficial comparison with our PQAG constructions. The original Muckle construction by Dowling et al. [50] integrates quantum key distribution (QKD) and relies on pre-shared symmetric keys for authentication, both of which are unsuitable for the specific constraints of the aviation domain. Muckle+ [34] enhances the original Muckle protocol by replacing pre-shared keys with digital signatures, while Muckle# [16] introduces implicit authentication through the use of KEMs. We note that Muckle+ and Muckle# are concurrent and independent of our work. While PQAG-SIG and PQAG-KEM exhibit superficial similarities to Muckle+ and Muckle#, respectively, there are key distinctions. Notably, Muckle+ and Muckle# require multiple round trips between the Initiator and Responder, whereas our constructions con-

fine key negotiation to a single round trip. Additionally, our approach incorporates only classical and post-quantum key components, explicitly avoiding the use of QKD keying material. Unlike Muckle+ and Muckle#, we further demonstrate the practical applicability of our constructions by instantiating and implementing them with relevant cryptographic primitives.

6.3 Implementation of PQAG

In this section we discuss our instantiation and reference implementation of the PQAG protocols. Instantiating both approaches allowed us to evaluate their relative benefits and costs. We can see that there exists a trade-off between the two approaches: using signatures as the basis for authentication is more computationally expensive, but less costly in bandwidth. Moreover, placing both approaches in the context of packet sizes further demonstrates their relative practicality in the aviation domain.

We implement PQAG-KEM and PQAG-SIG in Python, and benchmarked their performance on a Raspberry Pi to demonstrate practicality on constrained devices (and for uniform comparisons with previously existing protocols), and a standard desktop system. We compare our results with existing works [22, 86]. We modified the STS-SIDH protocol [86] with STS-CSIDH, due to security weaknesses of SIDH [41], using the post-quantum `sibc` library for CSIDH [104] and Ed25519 for digital signatures. It must be noted that the work of Bellido-Manganell et al. [22] does not provide performance metrics, only network performance results. We begin by discussing our choices of instantiations for cryptographic primitives.

Instantiation

PQAG aims for 128-bit post-quantum security against a quantum-equipped attacker. The efficiency of PQAG within resource-constrained environments was a critical consideration during instantiation. First, as a baseline, we instantiate a fully classical variant of PQAG as FCAG-SIG with ECC-DH as the classical KEM along with the classical signature scheme EdDSA. We instantiate PQAG-SIG with Kyber as the post-quantum KEM and Dilithium as the digital signature (which we denote PQAG-SIG-Ky-Di). To compare with classical signatures, we instantiate PQAG-SIG with either McEliece (PQAG-SIG-Mc) or Kyber (PQAG-SIG-Ky) as the underlying post-quantum KEM, but with classical signature scheme

EdDSA. For all three variants, the final derived shared keys were 256-bits long. Thus our choices of cryptographic algorithms for PQAG-SIG are:

- Classic CKEM: Elliptic-curve DH key exchange using curve P384 [70].
- Post-Quantum PQKEM: McEliece with parameter set 348864f [122] and Kyber-512 [122], both achieving 128-bit quantum security.
- SIG: EdDSA using curve P-384 [70] For PQAG-SIG-Ky-Di with NIST level 2 security claimed [100, 81].
- MAC: HMAC-SHA-256 [75] using 256-bit keys.
- KDF: HKDF-SHA-256 [74] using 256-bit keys.

We instantiate PQAG-KEM with Kyber as the post-quantum KEM (which we denote PQAG-KEM-Hy). To check how much our hybrid approach impacts performance, we also implemented a variant of PQAG-KEM that does not include any of the CKEM operations, instead simply performing PQKEM steps, which we denote PQAG-KEM-FQ. For both variants, the final derived shared keys were 256-bits long. Thus our choices of cryptographic algorithms for PQAG-KEM are:

- Classic CKEM: Elliptic-curve DH key exchange using curve P384 [70].
- Post-Quantum PQKEM: Kyber-512 [122], achieving 128-bit quantum security.
- MAC: HMAC-SHA-256 [75] using 256-bit keys.
- KDF: HKDF-SHA-256 [74] using 256-bit keys.

Implementation

We require the use of the Python `cryptography` library [122] for implementing CKEM and KDF, and `PyNaCl` [121] libraries for SIG cryptographic primitives. For PQAG-SIG-Ky, PQAG-SIG-Mc and PQAG-SIG-Ky-Di we require the use of the Python `pqcrypto` [100] library.

6.3.1 PQAG-SIG Computational Costs

We now profile the performance of the underlying cryptographic functions in terms of average execution times (for 100 iterations per cryptographic functions). Our benchmarking experiments are designed to provide a comparative evaluation of the PQAG protocols among their different initiations as well as existing literature, and also evaluate the cost of different post-quantum algorithms used in PQAG-SIG and PQAG-KEM respectively. Table 6.1

compares the performance of the cryptographic components of FCAG-SIG, PQAG-SIG-Mc, PQAG-SIG-Ky and PQAG-SIG-Ky-Di when run on two separate testbeds. Our experiments were performed on a Raspberry Pi 3 B+ running Raspberry Pi OS with a 1.4GHz quad core and 1GB RAM; and an Intel Core 1.80GHz i7-10510U CPU with 16GB RAM, running Windows 10 Home.

Operation (A)	FCAG **Pi**	PQAG McE **Pi**	PQAG Ky **Pi**	PQAG Di **Pi**	FCAG **Intel**	PQAG McE **Intel**	PQAG Ky **Intel**	PQAG Di **Intel**
PQKEM.KGen	N/A	1.1530	0.0015	0.0007	N/A	0.2814	0.0006	0.0002
CKEM.KGen	0.0028	0.0178	0.0268	0.0201	0.0003	0.0036	0.0027	0.0011
SIG.Sign	0.0018	0.0051	0.0024	0.0072	0.0002	0.0020	0.0003	0.0003
SIG.Vfy	0.0017	0.0058	0.0014	0.0015	0.0003	0.0021	0.0003	0.0001
KEM.Decaps	0.0045	0.0216	0.0191	0.0190	0.0006	0.0044	0.0028	0.0012
MAC	0.0001	0.0053	0.0001	0.0002	0.00002	0.0013	0.0001	0.0001
Operation (G)	FCAG **Pi**	PQAG McE **Pi**	PQAG Ky **Pi**	PQAG Di **Pi**	FCAG **Intel**	PQAG McE **Intel**	PQAG Ky **Intel**	PQAG Di **Intel**
CKEM.KGen	0.0082	0.0177	0.0205	0.0194	0.0004	0.0035	0.0027	0.0040
SIG.Sign	0.0022	0.0056	0.0011	0.0056	0.0002	0.0022	0.0003	0.0011
SIG.Vfy	0.0022	0.0054	0.0019	0.0016	0.0003	0.0022	0.0003	0.0004
KEM.Encaps	0.0101	0.0194	0.0193	0.0192	0.0005	0.0046	0.0031	0.0047
MAC	0.0007	0.0047	0.0002	0.0002	0.00002	0.0013	0.0001	0.0001

Table 6.1: Performance evaluation for cryptographic primitives (in seconds) on ***Raspberry Pi 3 B+*** *(left-hand side) and on* ***Intel Core i7-10510U CPU @ 1.80GHz.*** *(right-hand side).* KEM.Encaps *and* KEM.Decaps *combines* PQKEM, CKEM *and* KDF *operations into a single function. Note that* McE, Ky *and* Di *refer to* PQAG-SIG-Mc, PQAG-SIG-Ky *and* PQAG-SIG-Ky-Di, *respectively, and* A *contains results for aircraft operations and* G *results for ground stations.*

Our instantiation of FCAG-SIG had a slight edge in performance compared to PQAG-SIG-Ky-Di. This was due to FCAG-SIG having fewer operations to perform combined with the benefit of smaller key sizes of ECC-DH. Both PQAG-SIG-Ky and PQAG-SIG-Ky-Di achieve significantly better performance than PQAG-SIG-Mc, and as expected the desktop testbed was better for all performance metrics than the Raspberry Pi testbed. In general, the McEliece key generation had the highest average time per operation for both the test environments. Kyber key generation, on the other hand, was faster even when compared to the classical ECC-DH key generation. In addition, the SIG.Sign and SIG.Vfy operations of PQAG-SIG-Ky outperformed PQAG-SIG-Mc, due to the drastically smaller public keys.

This also applies to the MAC execution on both A and G: due to the smaller key sizes, the MAC operations were significantly faster for PQAG-SIG-Ky. Moreover, the SIG.Vfy of PQAG-SIG-Ky-Di was either on a par with that of PQAG-SIG-Ky and FCAG-SIG or performed slightly better owing to the fast verification times of Dilithium. In comparison to other operations, MAC execution on both A and G for both testbeds were significantly faster for all instantiations of PQAG and FCAG-SIG. Additionally, PQAG-SIG-Ky's SIG.Sign operations performed faster than PQAG-SIG-Ky-Di's due to its smaller key sizes. However, for both the instantiations, the difference in SIG.Vfy performance was negligible, confirming the faster verification times of Dilithium. For the desktop testbed, all cryptographic operations averaged well under a single second, whereas in the the Raspberry Pi 3 B+ testbed, all operations except the McEliece key generation averaged under a second. These results are promising for practical integration of PQAG-SIG-Ky into the constrained environments used in aviation infrastructure.

6.3.2 PQAG-KEM Computational Costs

Tables 6.2 compares the computational overhead of each cryptographic primitive of PQAG-KEM-Hy, PQAG-KEM-FQ when run on our two testbeds.

As shown in Table 6.2. It is clear that PQAG-KEM-Hy has higher computational overhead, due to the additional CKEM operations and key derivation steps. However, it is clear that the computational performance of the two instantiations is negligibly different, and we argue that the added hybrid layer of security in PQAG-KEM-Hy justifies the slight increase in computation time.

6.3.3 PQAG Performance Comparison

In Table 6.3 we compare the performance of all instantiations of PQAG against STS-SIDH [86]. For all protocols we profiled the walltime performance of the entire A and G execution (for 100 executions of FCAG-SIG, PQAG-KEM, PQAG-SIG and STS-SIDH). Both A and G are running on the same machine, and communication is exchanged via `localhost`.

In general STS-CSIDH proved computationally most expensive, averaging at significantly higher performance times in both testbeds. Particularly, within the Raspberry Pi testbed STS-CSIDH struggled to perform, and with occasionally lengthy "hang" times between subsequent operations and periodic restarts. The original STS-SIDH [86] is instantiated with SIDH which is around 10x times faster compared to CSIDH. However, since the publi-

Operation (A)	PQAG Hy **Pi**	PQAG FQ **Pi**	PQAG Hy **Intel**	PQAG FQ **Intel**
$\mathsf{PQKEM.KGen_E}$	0.00090	0.00115	0.00028	0.00028
$\mathsf{CKEM.KGen_E}$	0.00301	NA	0.00068	NA
$\mathsf{PQKEM.Encaps_G}$	0.00127	0.00243	0.00064	0.00029
$\mathsf{CKEM.Encaps_G}$	0.00313	NA	0.00082	NA
$\mathsf{CKEM.Decaps_E}$	0.00292	NA	0.00066	NA
$\mathsf{PQKEM.Decaps_A}$	0.00133	0.00160	0.00021	0.000092
$\mathsf{PQKEM.Decaps_E}$	0.00270	0.00135	0.00058	0.00011
MAC	0.00003	0.000061	0.00008	0.00007
Operation (G)	PQAG Hy **Pi**	PQAG FQ **Pi**	PQAG Hy **Intel**	PQAG FQ **Intel**
$\mathsf{CKEM.Encaps_E}$	0.00579	NA	0.00129	NA
$\mathsf{PQKEM.Encaps_E}$	0.00144	0.00090	0.00069	0.00019
$\mathsf{CKEM.Decaps_G}$	0.00293	NA	0.00068	NA
$\mathsf{PQKEM.Encaps_A}$	0.00124	0.00089	0.00065	0.00018
$\mathsf{PQKEM.Decaps_G}$	0.00321	0.00190	0.00069	0.00017
MAC	0.00003	0.000061	0.00010	0.00007

Table 6.2: Performance evaluation for cryptographic primitives used in PQAG-SIG *variants (in seconds) on* ***Raspberry Pi 3 B+*** *(left-hand side) and* ***Intel Core i7-10510U CPU @ 1.80GHz*** *(right-hand side). Note that* Hy *and* FQ *refers to the (hybrid secure)* PQAG-SIG-Ky*, and (purely post-quantum)* PQAG-KEM-FQ*, respectively, and* A *contains results for aircraft operations and* G *results for ground stations.*

Testbed	STS CSIDH	PQAG KEM FQ	PQAG KEM Hy	PQAG SIG McE	PQAG SIG Ky	PQAG SIG Di	FCAG SIG
Pi	312.8312	0.0103	0.0300	1.2513	0.0169	0.0947	0.0344
Intel	57.5576	0.0013	0.0087	0.3059	0.0024	0.0133	0.0028

Table 6.3: Comparison of walltime executions (in seconds).

cation of [86], SIDH has been proven insecure [41] and thus we instantiated STS-CSIDH [86] with CSIDH to guarantee its post-quantum security while maintaining its SIDH basis. Furthermore, STS-CSIDH requires an additional message from the ground station to the aircraft, adding to communication latency. In comparison PQAG-SIG-Mc fared significantly better in both testbeds, even when performance is affected by the large KEM public keys. Overall, among all the PQAG-SIG instantiations, PQAG-SIG-Ky offered the most competitive performance in the constrained Raspberry Pi testbed. Between PQAG-SIG-Ky and PQAG-SIG-Ky-Di, PQAG-SIG-Ky performed better due to the faster signing operations and smaller signature sizes of EdDSA compared to Dilithium. Moreover, the performance of PQAG-SIG-Ky was on a par with our baseline classical instantiation FCAG-SIG, with only a slight edge in performance for FCAG-SIG within the Raspberry Pi testbed.

Between the two implementations of PQAG-KEM, PQAG-KEM-FQ had a slight edge in performance, compared to PQAG-KEM-Hy. This was due to the fact that PQAG-KEM-FQ only uses post-quantum Kyber for all KEMs, whereas for PQAG-KEM-Hy we use hybrid KEMs, combining Kyber with ECC-DH to provide hybrid security. We conclude that the negligible increase in the computation times of PQAG-KEM-Hy is offset by the additional layer of security. Figures 6.4 and 6.5 visualize these findings. We chose to compare the walltime execution of our STS-CSIDH and PQAG-SIG-Mc separately in Figure 6.5 as their results drastically deviated from our other instantiations.

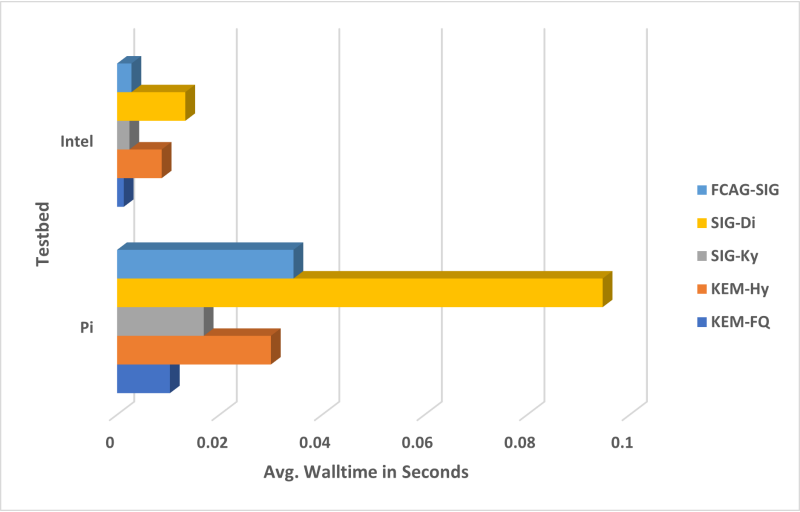


Figure 6.4: Walltime executions for FCAG-SIG, PQAG-SIG-Ky, PQAG-SIG-Ky-Di, PQAG-KEM-Hy *and* PQAG-KEM-FQ.

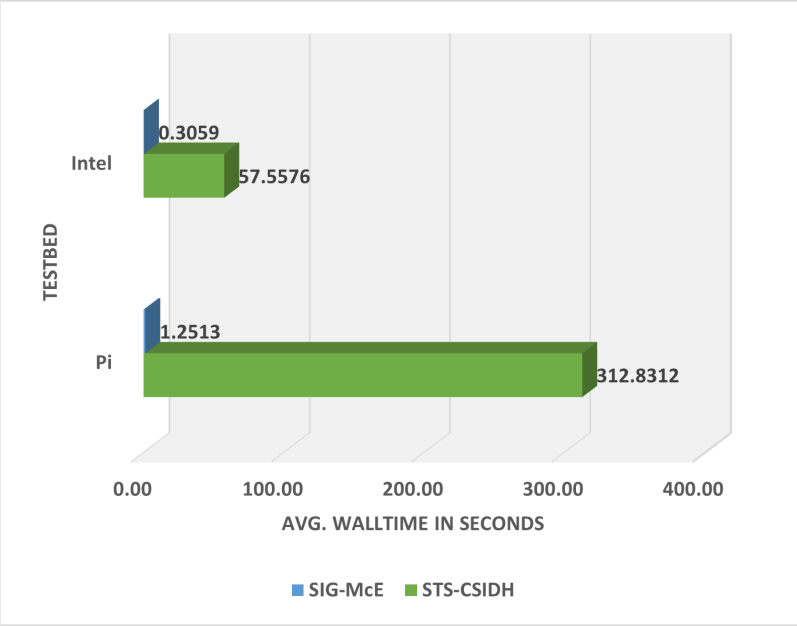


Figure 6.5: Walltime executions for STS-CSIDH *and* PQAG-SIG-Mc.

Communication Cost

For all our instantiations, Table 6.4 and Figure 6.6 compare the transmission times per round-trip, for all frequently used data-links in ground-to-air avionic communication. This was calculated by dividing the packet size of each instantiation (Table 6.5) by the expected bandwidth for each medium. For each calculated transmission time we have an additional Δ value to account for various factors affecting the round-trip time, such as latency. This Δ value is controlled by the specifics constraints of the communication links and the setting in which its used. For instance, the expected Δ of STS-CSIDH is extremely high due to the long computational times ($10\times$ slower than the now defunct SIDH). For PQAG-McEliece, the Δ unacceptably increases due to the large public keys (261,120 bytes). Satellite data-links have a longer RTT due to the distance between nodes, which in turn will affect the Δ.

Table 6.6 compares the communication complexity of all our PQAG instantiations. We consider the length of the cryptographic components of each message, in particular we do not capture header sizes, nor nonce or *id* values, as they are either constant or setting-specific. The McEliece public key is the most bandwidth-consuming component of PQAG-SIG-Mc at

Protocol / Datalink	FCAG	STS CSIDH	PQAG KEM FQ	PQAG KEM Hy	PQAG SIG McE	PQAG SIG Ky	PQAG SIG Di
VDLm2 31.5kbps	0.166+Δ	0.654+Δ	0.998+Δ	1.170+Δ	66.940+Δ	0.560+Δ	1.554+Δ
AeroMACS 1.8-9.2Mbps	0.003 - 0.0006+Δ	0.001 - 0.0002+Δ	0.018 - 0.003+Δ	0.0201 - 0.004+Δ	1.164 - 0.2274+Δ	0.01 - 0.0019+Δ	0.028- 0.006+Δ
LDACS 0.6-2.8Mbps	0.0088 - 0.0018+Δ	0.003 - 0.0007+Δ	0.052 - 0.011+Δ	0.062 - 0.013+Δ	3.492 - 0.7483+Δ	0.029 - 0.0062+Δ	0.082 - 0.018+Δ
Inmarsat SB 432kbps	0.0122+Δ	0.005+Δ	0.072+Δ	0.085+Δ	4.85+Δ	0.041+Δ	0.113+Δ
Iridiusm Certus 22-704kbps	0.2378 - 0.0074+Δ	0.094 - 0.003+Δ	1.42 - 0.044+Δ	1.665 - 0.0521+Δ	95.237 - 2.976+Δ	0.797 - 0.025+Δ	2.225 - 0.07+Δ

Table 6.4: Comparison of transmission times (in seconds) per round-trip. Δ represents any additional latency added by the constraints of the communication link.

Implementation	Hello	Response
STS-CSIDH	128	128
PQAG-KEM-FQ	2384	1520
PQAG-KEM-Hy	2740	1844
PQAG-SIG-Mc	261447	455
PQAG-SIG-Ky	1090	1026
PQAG-SIG-Ky-Di	3075	3043

Table 6.5: Hello and Response message sizes (in bytes) for each implementation.

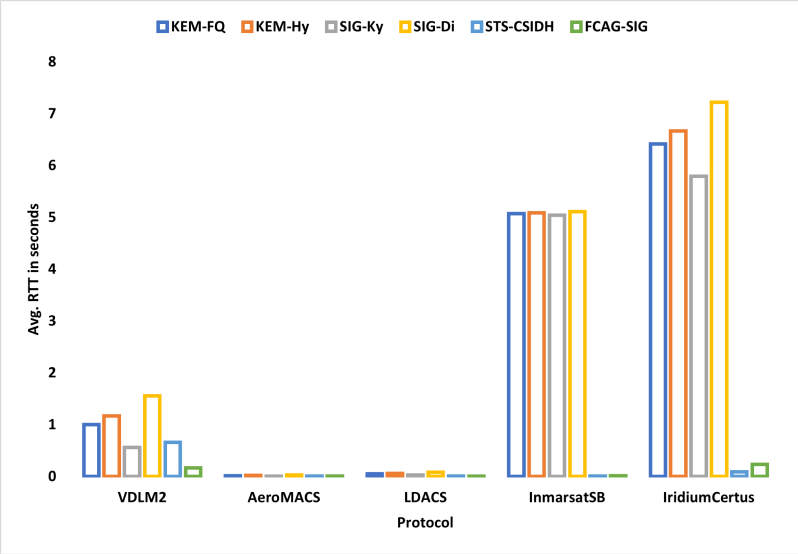


Figure 6.6: Comparison of transmission times (in seconds) per round-trip.

261Kb. However, the PQKEM ciphertext size of McEliece was smaller than that of Kyber, which may prove advantageous in certain circumstances, such as in the case of data-links with asymmetric bandwidths for forward-and-return links. Moreover, the signature size of Dilithium (2044B) is significantly larger compared to the classical counterpart EdDSA (96B), which will add to the communication overhead. Additionally, it must be noted that although the output of ECC-P384R1 is 384-bits, due to additional encodings of the Python library used, the resulting final output produces ECC keys that are 215-bytes long, which can be optimized.

Remark 6. *Standards in avionic communications focus on bandwidth constraints as opposed to packet size. However, the packet size of the underlying communication protocol (LDACS) is 1400 bytes [22] and our maximum packet size from* PQAG-SIG-Ky *is 1090 bytes. Even if we exchange signature public keys,* PQAG-SIG-Ky *achieves messages that fit within this packet size. This allows us to recommend a specific instantiation from our* PQAG *framework that fits within a single LDACS packet using NIST PQC approved components (Kyber), thus preventing unnecessary packet fragmentation. Our next closest full-quantum instantiation is* PQAG-KEM-FQ *at maximum 2384, which fits within two packets.*

Component	FCAG	PQAG-SIG McE	PQAG-SIG Ky	PQAG-SIG Di	PQAG-KEM FQ	PQAG-KEM Ky
PQKEM *pqpk*	N/A	261120	800	800	800	800
PQKEM *pqctxt*	N/A	112	736	736	736	736
CKEM *cpk*	215	215	215	215	N/A	215
SIG σ	96	96	96	2044	N/A	N/A
MACτ	32	32	32	32	32	32

Table 6.6: Communication cost of the underlying cryptographic components of PQAG-KEM *and* PQAG-KEM *respectively (in bytes).*

6.3.4 Storage Costs

PQAG-SIG does not assume pre-distribution of KEM public keys, but does assume pre-distribution of signature public keys. For each protocol, the storage cost of the A is as follows (assuming x interactions with independent ground stations G):

- PQAG-KEM-FQ: 800x bytes
- PQAG-KEM-Hy: 1015x bytes
- FCAG-SIG: 215x bytes
- PQAG-SIG-Ky: 215x bytes
- PQAG-SIG-Mc: 215x bytes
- PQAG-SIG-Ky-Di: 1527x bytes.

6.4 Security Analysis

In this section we analyze our proposed PQAG-SIG and PQAG-KEM protocols, by utilizing the simplified HAKE model presented in Section 5.3. We begin by presenting our first result, the security of PQAG-SIG, in Theorem 8.

Theorem 8 (PQAG-SIG Security)**.** *The* PQAG-SIG *protocol presented in Section 6.2 is post-quantum secure under cleanness predicate* $\mathsf{clean}_{q\mathsf{HAKE}}$ *(capturing perfect forward security and resilience to KCI attacks against $\mathcal{A}$). That is, for any QPT algorithm $\mathcal{A}$ against the key-indistinguishability game (defined in Figure 5.2),* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* dual-prf*,* ind-cca*,* sufcma *and* sufcma *security of the* PRF*,* PRF*,* KEM*,* MAC *and*

SIG *primitives respectively. Thus we have:* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca},\mathcal{A}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{MAC},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{SIG},n_P,n_S}(\lambda)$.

Proof. We now turn to proving our result. We split our analysis into three mutually-exclusive cases:

1. **Case 1** assumes that the test session π^s_i (such that $\mathcal{A}$ issued $\mathsf{Test}(i,s)$) is an initiator session, and that π^s_i has no *matching partner* (as in Figure 5.3). We define the QPT algorithm $\mathcal{A}$'s advantage in **Case 1** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$.
2. **Case 2** assumes that the test session π^s_i is a responder , and that π^s_i has no *matching partner*. We define $\mathcal{A}$'s advantage in **Case 2** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$.
3. **Case 3** assumes that the test session π^s_i has a *matching partner*. We define $\mathcal{A}$'s advantage in **Case 3** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$.

It is clear that: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$, thus we bound $\mathcal{A}$'s advantage in each case separately.

In **Case 1** and **Case 2** we show that $\mathcal{A}$'s advantage in causing the test session π^s_i to accept without a matching partner is negligible, and thus the $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible (since the experiment does not differ based on the challenge bit b, as the π^s_i does not compute a real-or-random session key).

In **Case 3** we replace the computation of the real session key by the test session π^s_i with a uniformly random key. Thus, the distribution of the keys returned by π^s_i are identical, regardless of the value of the challenge bit b, and we can show that $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible. We now begin with the first case.

Case 1: Test `init` session without origin session

We begin by showing that $\mathcal{A}$ has negligible chance in causing π^s_i to reach an `accept` state without a matching session. We do so via the following sequence of game hops:

Game 0 This is the HAKE security game and $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) = \Pr(break_0)$.

Game 1 In this game, we guess the index (i,s) and the intended partner j and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi^{s'}_{i'}$ such that $\pi^{s'}_{i'}.pid = j'$ and $(i,s,j) \neq (i',s',j')$. Thus: $\Pr(break_0) \leq {n_P}^2 \cdot n_S \cdot \Pr(break_1)$.

Game 2 In this game we abort if the test session π^s_i sets the status $\pi^s_i.\alpha \leftarrow$ `reject`. Note that by the previous game we abort if the Test query is issued to a session that is not

π_i^s, and if $\pi_i^s.pid \neq j$. If the session π_i^s ever reaches the status $\pi_i^s.\alpha \leftarrow \texttt{reject}$, then the challenger will respond to the $\mathsf{Test}(i,s)$ query with $\perp$, and thus the difference in $\mathcal{A}$'s advantage between **Game 2** and **Game 3** is 0. Thus: $\Pr(break_1) \leq \Pr(break_2)$.

Game 3 In this game we define an abort event $\mathbf{abort}_\alpha$ that triggers if the test session π_i^s sets the status $\pi_i^s.\alpha \leftarrow \texttt{accept}$. We note that the response to $\mathsf{Test}(i,s)$ issued by $\mathcal{A}$ is always $\perp$, (since the challenger aborts the game if π_i^s accepts, and $\mathsf{Test}(i,s) = \perp$ when π_i^s rejects the protocol execution), and thus $\Pr(break_3) = 0$. In **Game 4** and **Game 5** we prove that the probability of $\mathcal{A}$ in causing $\mathbf{abort}_\alpha$ to trigger is negligible. Thus: $\Pr(break_2) \leq \Pr(\mathbf{abort}_\alpha)$.

Game 4 In this game we abort if the test session π_i^s receives a signature σ_1 (signed over $m_0 \| m_1$) that verifies correctly but there exists no honest session π_j^t that has output σ_1. Specifically, in **Game 4** we define a reduction $\mathcal{B}_1$ against the sufcma security of the signature scheme SIG. At the beginning of the experiment, when $\mathcal{B}_1$ receives the list of public-keys $(pk_1, \ldots, pk_{n_P})$ from $\mathcal{C}$, $\mathcal{B}_1$ initializes a sufcma challenger $\mathcal{C}_{\mathsf{sufcma}}$, and replaces pk_j with pk output by $\mathcal{C}_{\mathsf{sufcma}}$. Whenever $\mathcal{A}$ initializes a session owned by P_j, then $\mathcal{B}_1$ generates m_0 as usual, but queries $\mathcal{C}_{\mathsf{sufcma}}$ with m_0 to get a signature σ_0 over m_0. Similarly, whenever $\mathcal{A}$ issues $\mathsf{Send}(j,t,m)$ to a session π_j^t owned by P_j, then $\mathcal{B}_1$ verifies m and computes m_1 as usual, but queries $\mathcal{C}_{\mathsf{sufcma}}$ with $m \| m_1$ to receive σ_1 over $m \| m_1$. These changes are indistinguishable to $\mathcal{A}$, as $\mathcal{C}_{\mathsf{sufcma}}$ generates public-keys and signatures identically to $\mathcal{C}$, so $\mathcal{A}$ cannot detect this replacement. Note that if $\mathcal{A}$ has issued a $\mathsf{CorruptQK}(j)$ query before π_i^s receives σ_1, then $\mathsf{clean}_{q\mathsf{HAKE}} = \texttt{false}$ for π_i^s, and $\mathcal{C}$ returns $b^* \xleftarrow{\$} \{0,1\}$ regardless of the challenge bit b. Thus, any $\mathcal{A}$ with non-negligible advantage must not yet have issued $\mathsf{CorruptQK}(j)$. Also, as the result of **Game 2** and **Game 3**, the game aborts after π_i^s receives m_1, σ_1, so $\mathcal{A}$ cannot later issue a $\mathsf{CorruptQK}(j)$ query.

By the definition of **Case 1**, π_i^s sets the status $\pi_i^s.\alpha \leftarrow \texttt{accept}$ despite there being no honest session that outputs m_1, σ_1. Thus, $\mathcal{B}_1$ never queried $m_0 \| m_1$ to $\mathcal{C}_{\mathsf{sufcma}}$, and it follows that m_1, σ_1 is a forged message. Thus, if π_i^s receives a signature σ_1 (signed over $m_0 \| m_1$) that verifies correctly but there exists no honest session π_j^t that has output σ_1, then $\mathcal{B}_1$ wins the sufcma security game against the signature scheme SIG, and $\Pr(\mathbf{abort}_\alpha) = \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{B}_1}(\lambda)$.

Since π_i^s now aborts when verifying σ_1, it cannot trigger $\mathbf{abort}_\alpha$ and thus we have: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq {n_P}^2 n_S \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{B}_1}(\lambda)$.

Case 2: Test responder session without origin session

We now show that $\mathcal{A}$ has negligible chance in causing π_i^s (with $\pi_i^s.\rho = \mathsf{responder}$) to reach an `accept` state without an origin session. As the proof of **Case 2** follows analogously to **Case 1** with a minor change in notation up to **Game 3**, we omit these game hops and proceed from **Game 4**.

Game 4 In this game we abort if the test session π_i^s receives a signature σ_0 (signed over m_0) that verifies correctly but there exists no honest session π_j^t that has output σ_0. Specifically, in **Game 4** we define a reduction $\mathcal{B}_2$ against the sufcma security of the signature scheme SIG. At the beginning of the experiment, when $\mathcal{B}_2$ receives the list of public-keys $(pk_1, \ldots, pk_{n_P})$ from $\mathcal{C}$, $\mathcal{B}_1$ initializes a sufcma challenger $\mathcal{C}_{\mathsf{sufcma}}$, and replaces pk_j with pk output by $\mathcal{C}_{\mathsf{sufcma}}$. Whenever $\mathcal{A}$ initializes a session owned by P_j, then $\mathcal{B}_2$ generates m_0 as usual, but queries $\mathcal{C}_{\mathsf{sufcma}}$ with m_0 to get a signature σ_0 over m_0. Similarly, whenever $\mathcal{A}$ issues $\mathsf{Send}(j, t, m)$ to a session π_j^t owned by P_j, then $\mathcal{B}_2$ verifies m and computes m_1 as usual, but queries $\mathcal{C}_{\mathsf{sufcma}}$ with $m \| m_1$ to receive σ_1 over $m \| m_1$. These changes are indistinguishable to $\mathcal{A}$, as $\mathcal{C}_{\mathsf{sufcma}}$ generates public-keys and signatures identically to $\mathcal{C}$, so $\mathcal{A}$ cannot detect this replacement. Note that if $\mathcal{A}$ has issued a $\mathsf{CorruptQK}(j)$ query before π_i^s receives σ_0, then $\mathsf{clean}_{q\mathsf{HAKE}} = \mathtt{false}$ for π_i^s, and $\mathcal{C}$ returns $b^* \xleftarrow{\$} \{0, 1\}$ regardless of the challenge bit b. Thus, any $\mathcal{A}$ with non-negligible advantage must not yet have issued $\mathsf{CorruptQK}(j)$. Also, as the result of **Game 2** and **Game 3**, the game aborts after π_i^s receives τ, so $\mathcal{A}$ cannot later issue a $\mathsf{CorruptQK}(j)$ query.

By the definition of the abort event, $\mathcal{B}_2$ never queried m_0 to $\mathcal{C}_{\mathsf{sufcma}}$, and it follows that m_0, σ_0 is a forged message. Thus, if π_i^s receives a signature σ_0 that verifies correctly but there exists no honest session π_j^t that has output σ_0, then $\mathcal{B}_2$ wins the sufcma security game against the signature scheme SIG, and $\Pr(\mathbf{abort}_\alpha) = \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{B}_2}(\lambda) + \Pr(break_4)$.

Game 5 In this game, we replace the key pqk_{E} derived in the test session π_i^s with the uniformly random and independent value $\widetilde{pqk}$. We define a reduction $\mathcal{B}_3$ that interacts with a $\mathsf{ind\text{-}cca}$ KEM challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$ (as described in Definition 23) who replaces the $pqpk_{\mathsf{E}}$ value sent in m_0 with the public-key pk received from $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. When potential partner π_j^t should compute the ciphertext $pqctxt_{\mathsf{E}}$ sent in m_1, $\mathcal{B}_3$ instead replaces the computation of $pqctxt_{\mathsf{E}}$ and pqk_{E} with the outputs of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, $\widetilde{pqctxt}_{\mathsf{E}}$ and $\widetilde{pqk}$ respectively. Whenever party j requires the use of the secret key to decapsulate

a ciphertext $pqctxt'_{\mathsf{E}} \neq \widetilde{pqctxt_{\mathsf{E}}}$, $\mathcal{B}_3$ simply queries $pqctxt'_{\mathsf{E}}$ to $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces the computation of pqk' with the output of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. By **Game 4**, we know that $pqpk_{\mathsf{E}}$ sent in m_0 must have been sent from an honest session π_j^t owned by P_j without modification. Any adversary that can detect the replacement of pqk_{E} with a uniformly random value $\widetilde{pqk}$ implies an efficient distinguishing algorithm $\mathcal{B}_3$ against the $\mathsf{ind\text{-}cca}$ security of KEM. Thus: $\Pr(break_4) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_3}(\lambda) + \Pr(break_5)$.

Game 6 In this game we replace the computation of the extracted key $ms = \mathsf{PRF}(ck_{\mathsf{E}}, \widetilde{pqk})$ with a uniformly random and independent value $\widetilde{ms} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_4$ that initializes a $\mathsf{dual\text{-}prf}$ challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ when π_i^s needs to compute $\mathsf{PRF}(ck_{\mathsf{E}}, \widetilde{pqk})$ and instead queries ck_{E} to $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. $\mathcal{B}_4$ uses the output of the query $\widetilde{ms}$ to replace the computation of ms. Since $\widetilde{pqk}$ is uniformly random and independent by **Game 5**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseQK}(i,s)$ or $\mathsf{CompromiseQK}(j,t)$ (since the communicating partner has sent a message m_0 that was received without modification by $\mathcal{A}$), this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 0, then $\widetilde{ms} = \mathsf{PRF}(ck_{\mathsf{E}}, \widetilde{pqk})$ and we are in **Game 5**. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 1, then $\widetilde{ms} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 6**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_4$ against the $\mathsf{dual\text{-}prf}$ security of PRF, and we find: $\Pr(break_5) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_4}(\lambda) + \Pr(break_6)$.

Game 7 In this game we replace the computation of the expanded keys $mk, k = \mathsf{PRF}(\widetilde{ms}, \epsilon)$ with a uniformly random and independent values $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_5$ that initializes a prf challenger $\mathcal{C}_{\mathsf{prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ms}, \epsilon)$ and instead queries ϵ to $\mathcal{C}_{\mathsf{prf}}$. $\mathcal{B}_5$ uses the output $\widetilde{mk}, \widetilde{k}$ to replace the computation of mk, k. Since $\widetilde{ms}$ is already uniformly random and independent by **Game 6**, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 0, then $\widetilde{mk}, \widetilde{k} = \mathsf{PRF}(\widetilde{ms}, \epsilon)$ and we are in **Game 6**. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 1, then $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 7**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful distinguishing adversary $\mathcal{B}_5$ against the prf security of PRF, and we find $\Pr(break_6) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_5}(\lambda) + \Pr(break_7)$.

Game 8 In this game we abort if the test session π_i^s receives a message τ (computed over

$m_0 \| m_1$) that verifies correctly but there exists no honest session π_j^t that has output τ. Specifically, in **Game 8** we define a reduction $\mathcal{B}_6$ against the sufcma security of the Message Authentication Code MAC. When $\mathcal{B}_6$ needs to compute a MAC over $m_0 \| m_1$ using $\widetilde{mk}$, $\mathcal{B}_6$ computes the MAC by initializing a sufcma challenger $\mathcal{C}_{\mathsf{sufcma}}$ and querying $m_0 \| m_1$. No changes to the experiment occur, as $\mathcal{C}_{\mathsf{sufcma}}$ computes MACs identically to $\mathcal{C}$, so $\mathcal{A}$ cannot detect this replacement. Also, as the result of **Game 7**, $\widetilde{mk}$ is a uniformly random and independent value, so this replacement is sound.
By the definition of **Case 2**, π_i^s sets the status $\pi_i^s.\alpha \leftarrow \texttt{accept}$ despite there being no honest session that matches π_i^s. Thus, $\mathcal{B}_6$ never queried $m_0 \| m_1$ to $\mathcal{C}_{\mathsf{sufcma}}$, and it follows that τ is a forged message. Thus, if π_i^s receives a MAC tag τ (over $m_0 \| m_1$) that verifies correctly but there exists no honest session π_j^t that matches π_i^s, then $(m_0 \| m_1, \tau)$ represents a valid forgery and $\mathcal{B}_6$ wins the sufcma security game against MAC, and $\Pr(break_8) = \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_6}(\lambda)$.
Since π_i^s now aborts when verifying τ, it cannot trigger $\mathbf{abort}_\alpha$ and thus:

$$\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq {n_P}^2 n_S \cdot \big(\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{SIG},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_4}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_6}(\lambda)\big).$$

We now complete our proof by bounding $\mathcal{A}$'s advantage in **Case 3**.

Case 3: Test session with matching session

In **Case 3**, we show that if $\mathcal{A}$ that has issued a $\mathsf{Test}(i,s)$ query to a clean session π_i^s, then $\mathcal{A}$ has negligible advantage in guessing the test bit b. In what follows, we note that for the cleanness predicate $\mathsf{clean}_{q\mathsf{HAKE}}$ to be upheld by π_i^s, then $\mathsf{CompromiseQK}(i,s)$, $\mathsf{CompromiseQK}(j,t)$ cannot be queried (where π_j^t *matches* π_i^s). Thus, we can assume in what follows that $\mathcal{A}$ has not compromised the *post-quantum ephemeral* KEM secrets. We now show that $\mathcal{A}$ has negligible advantage in guessing the test bit b, via the following series of game hops:

Game 0 This is the HAKE security game, and $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) = \Pr(break_0)$.

Game 1 In this game, we guess the index (i,s) and the matching session (j,t) and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi_{i'}^{s'}$ such that $\pi_{t'}^{j'}$ matches $\pi_{i'}^{s'}$ and $(i,s),(j,t) \neq (i',s'),(j',t')$. Thus $\Pr(break_0) \leq n_P^2 \cdot n_S^2 \cdot \Pr(break_1)$.

Game 2 This game proceeds identically to **Case 2 Game 5**, and we replace the key pqk_{E} derived in the test session π_i^s with the uniformly random and independent value $\widetilde{pqk}$ by defining a reduction $\mathcal{B}_7$ that interacts with a ind-cca KEM challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. By the definition of **Case 3**, we know that $pqpk_{\mathsf{E}}$ (or $pqctxt_{\mathsf{E}}$, respectively) sent in m_0 (resp. m_1) must have been sent from an honest session π_j^t owned by P_j without modification if $\pi_i^s.\rho = \texttt{resp}$ (resp. $\texttt{init}$). Any adversary that can detect the replacement of pqk_{E} with a uniformly random value $\widetilde{pqk}$ implies an efficient distinguishing algorithm $\mathcal{B}_7$ against the ind-cca security of KEM. Thus: $\Pr(break_1) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_7}(\lambda) + \Pr(break_2)$.

Game 3 In this game we replace the computation of the extracted key $ms = \mathsf{PRF}(ck_{\mathsf{E}}, \widetilde{pqk})$ with a uniformly random and independent value $\widetilde{ms} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_8$ that initializes a dual-prf challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ when π_i^s needs to compute $\mathsf{PRF}(ck_{\mathsf{E}}, \widetilde{pqk})$ and instead queries ck_{E} to $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. $\mathcal{B}_8$ uses the output of the query $\widetilde{ms}$ to replace the computation of ms. Since $\widetilde{pqk}$ is uniformly random and independent by **Game 2**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseQK}(i,s)$ or $\mathsf{CompromiseQK}(j,t)$, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 0, then $\widetilde{ms} = \mathsf{PRF}(ck_{\mathsf{E}}, \widetilde{pqk})$ and we are in **Game 2**. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 1, then $\widetilde{ms} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 3**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_8$ against the dual-prf security of PRF, and we find: $\Pr(break_2) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_8}(\lambda) + \Pr(break_3)$.

Game 4 In this game we replace the computation of the expanded keys $mk, k = \mathsf{PRF}(\widetilde{ms}, \epsilon)$ with a uniformly random and independent values $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and its matching session π_j^t. We define a reduction $\mathcal{B}_9$ that initializes a prf challenger $\mathcal{C}_{\mathsf{prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ms}, \epsilon)$ and instead queries ϵ to $\mathcal{C}_{\mathsf{prf}}$. $\mathcal{B}_9$ uses the output $\widetilde{mk}, \widetilde{k}$ to replace the computation of mk, k. Since $\widetilde{ms}$ is already uniformly random and independent by **Game 3**, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 0, then $\widetilde{mk}, \widetilde{k} = \mathsf{PRF}(\widetilde{ms}, \epsilon)$ and we are in **Game 3**. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 1, then $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 4**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful distinguishing adversary $\mathcal{B}_9$ against the prf security of PRF, and we find $\Pr(break_3) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_9}(\lambda) + \Pr(break_4)$. Since $\widetilde{k}$ is now uniformly random and

independent of the protocol flow regardless of the test bit b, $\mathcal{A}$ has no advantage in guessing the test bit and thus:
$\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq {n_P}^2 {n_S}^2 \cdot \left(\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_7}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_8}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_9}(\lambda)\right)$.

□

Next we present the security of PQAG-SIG against purely classical adversaries, i.e. $\mathcal{A}$ is a PPT algorithm, with cleanness predicate $\mathsf{clean}_{c\mathsf{HAKE}}$ in Theorem 9.

Theorem 9 (PQAG-SIG Classical Security)**.** *The* PQAG-SIG *protocol presented in Section 6.2 is secure under cleanness predicate* $\mathsf{clean}_{c\mathsf{HAKE}}$ *(capturing perfect forward security and resilience to KCI attacks against a classical adversary $\mathcal{A}$). That is, for any PPT algorithm $\mathcal{A}$ against the key-indistinguishability game (defined in Figure 5.2),* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$ *is negligible under the* dual-prf*,* prf*,* ind-cca*,* sufcma *and* sufcma *security of the* PRF*,* PRF*,* KEM*,* MAC *and* SIG *primitives respectively. Thus we have:* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca},\mathcal{A}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{MAC},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{SIG},n_P,n_S}(\lambda)$.

Proof. We now turn to proving our result. We split our analysis into three mutually-exclusive cases:

1. **Case 1** assumes that the test session π^s_i (such that $\mathcal{A}$ issued $\mathsf{Test}(i,s)$) is an initiator session, and that π^s_i has no *matching partner* (as in Figure 5.3). We define the PPT algorithm $\mathcal{A}$'s advantage in **Case 1** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$.

2. **Case 2** assumes that the test session π^s_i is a responder , and that π^s_i has no *matching partner*. We define $\mathcal{A}$'s advantage in **Case 2** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$.

3. **Case 3** assumes that the test session π^s_i has a *matching partner*. We define the PPT algorithm $\mathcal{A}$'s advantage in **Case 3** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$.

It is clear that: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A}}_{\mathsf{KEX},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda)$, thus we bound $\mathcal{A}$'s advantage in each case separately.

In **Case 1** and **Case 2** we show that $\mathcal{A}$'s advantage in causing the test session π^s_i to accept without a matching partner is negligible, and thus the $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible (since the experiment does not differ based on the challenge bit b, as the π^s_i does not compute a real-or-random session key). As the

analysis of **Case 1** and **Case 2** follows identically the corresponding analysis of Theorem 8, we omit the details here, and point the reader to the formal proofs for 8.

In **Case 3** we replace the computation of the real session key by the test session π_i^s with a uniformly random key. Thus, the distribution of the keys returned by π_i^s are identical, regardless of the value of the challenge bit b, and we can show that $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible. We now begin with the third case.

Case 3: Test session with matching session

In **Case 3**, we show that if $\mathcal{A}$ that has issued a $\mathsf{Test}(i,s)$ query to a clean session π_i^s, then $\mathcal{A}$ has negligible advantage in guessing the test bit b. In what follows, we split our analysis of Case 3 into the following sub-cases, each corresponding to a condition necessary for the cleanness predicate $\mathsf{clean}_{c\mathsf{HAKE}}$ to be upheld by π_i^s. These are the subcases (where π_j^t *matches* π_i^s):

- **Subcase 3.1**: $\mathsf{CompromiseQK}(i,s)$, $\mathsf{CompromiseQK}(j,t)$ were not queried.
- **Subcase 3.2**: $\mathsf{CompromiseCK}(i,s)$, $\mathsf{CompromiseCK}(j,t)$ were not queried.

It is straightforward to see that the advantage of $\mathcal{A}$ in Case 3 is bound by the sum of the advantages of $\mathcal{A}$ in all subcases. It is also straightforward to see that the proof of **Subcase 3.1** follows identically the corresponding analysis of **Case 3** of Theorem 8, and so we omit the details here, and point the reader to formal proofs for Theorem 8. We now treat the second subcase (**Subcase 3.2**), where $\mathcal{A}$ has not compromised the *classic ephemeral* KEM secrets.

3.2: $\mathsf{CompromiseCK}(i,s)$, $\mathsf{CompromiseCK}(j,t)$ have not been issued, where π_j^t *matches* π_i^s.

Game 0 This is the HAKE security game, and $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C3.2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) = \Pr(break_0)$.

Game 1 In this game, we guess the index (i,s) and the matching session (j,t) and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi_{i'}^{s'}$ such that $\pi_{t'}^{j'}$ matches $\pi_{i'}^{s'}$ and $(i,s),(j,t) \neq (i',s'),(j',t')$. Thus $\Pr(break_0) \leq n_P^2 \cdot n_S^2 \cdot \Pr(break_1)$.

Game 2 In this game, we replace the key ck_{E} derived in the test session π_i^s with the uniformly random and independent value $\widetilde{ck}$. We define a reduction $\mathcal{B}_{10}$ that interacts with a $\mathsf{ind\text{-}cca}$ KEM challenger (as described in Definition 23) and replaces the

cpk_{E} value sent in m_0 with the public-key pk received from $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. When potential partner should compute the ciphertext $cctxt_{\mathsf{E}}$ sent in m_1, $\mathcal{B}_{10}$ instead replaces the computation of $cctxt_{\mathsf{E}}$ and ck_{E} with the outputs of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, $\widetilde{cctxt_{\mathsf{E}}}$ and $\widetilde{ck}$ respectively. Whenever party j requires the use of the secret key to decapsulate a ciphertext $cctxt'_{\mathsf{E}} \neq \widetilde{cctxt_{\mathsf{E}}}$, $\mathcal{B}_{10}$ simply queries $cctxt'_{\mathsf{E}}$ to $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces the computation of ck' with the output of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. By the definition of **Case 3**, we know that cpk_{E} (or $cctxt_{\mathsf{E}}$, respectively) sent in m_0 (resp. m_1) must have been sent from an honest session π_j^t owned by P_j without modification if $\pi_i^s.\rho = \texttt{resp}$ (resp. $\texttt{init}$). Any adversary that can detect the replacement of ck_{E} with a uniformly random value $\widetilde{ck}$ implies an efficient distinguishing algorithm $\mathcal{B}_{10}$ against the $\mathsf{ind\text{-}cca}$ security of KEM. Thus: $\Pr(break_1) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_{10}}(\lambda) + \Pr(break_2)$.

Game 3 In this game we replace the computation of the extracted key $ms = \mathsf{PRF}(\widetilde{ck}, pqk_{\mathsf{E}})$ with a uniformly random and independent value $\widetilde{ms} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_{11}$ that initializes a prf challenger $\mathcal{C}_{\mathsf{prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ck}, pqk_{\mathsf{E}})$ and instead queries pqk_{E} to $\mathcal{C}_{\mathsf{prf}}$. $\mathcal{B}_{11}$ uses the output of the query $\widetilde{ms}$ to replace the computation of ms. Since $\widetilde{ck}$ is uniformly random and independent by **Game 2**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseCK}(i,s)$ or $\mathsf{CompromiseCK}(j,t)$, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 0, then $\widetilde{ms} = \mathsf{PRF}(\widetilde{ck}, pqk_{\mathsf{E}})$ and we are in **Game 2**. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 1, then $\widetilde{ms} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 3**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_{11}$ against the prf security of PRF, and we find: $\Pr(break_2) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{11}}(\lambda) + \Pr(break_3)$.

Game 4 In this game we replace the computation of the expanded keys $mk, k = \mathsf{PRF}(\widetilde{ms}, \epsilon)$ with a uniformly random and independent values $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and its matching session π_j^t. We define a reduction $\mathcal{B}_{12}$ that initializes a prf challenger $\mathcal{C}_{\mathsf{prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ms}, \epsilon)$ and instead queries ϵ to $\mathcal{C}_{\mathsf{prf}}$. $\mathcal{B}_{12}$ uses the output $\widetilde{mk}, \widetilde{k}$ to replace the computation of mk, k. Since $\widetilde{ms}$ is already uniformly random and independent by **Game 3**, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 0, then $\widetilde{mk}, \widetilde{k} = \mathsf{PRF}(\widetilde{ms}, \epsilon)$ and we are in **Game 3**. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 1, then $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in

Game 4. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful distinguishing adversary $\mathcal{B}_{12}$ against the prf security of PRF, and we find $\Pr(break_3) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{12}}(\lambda) + \Pr(break_4)$. Since $\widetilde{k}$ is now uniformly random and independent of the protocol flow regardless of the test bit b, $\mathcal{A}$ has no advantage in guessing the test bit and thus:
$\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C3.2}}_{\mathsf{PQAG\text{-}SIG},n_P,n_S}(\lambda) \leq {n_P}^2{n_S}^2 \cdot \big(\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_{10}}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{11}}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{12}}(\lambda)\big)$.

□

Now we turn to proving the security of PQAG-KEM. We note that the proof of PQAG-KEM follows closely the proof of Theorem 8, with minor changes in Case 1 and Case 2, where we demonstrate that a session will not accept without a matching partner due to the use of post-quantum KEMs. As such, we mainly detail the proof of Case 1, and refer the reader to 8 for detailed proof.

Theorem 10 (PQAG-KEM Security)**.** *The* PQAG-KEM *protocol presented in Section 6.2 is post-quantum secure under cleanness predicate* $\mathsf{clean}_{q\mathsf{HAKE}}$ *(capturing perfect forward security and resilience to KCI attacks against $\mathcal{A}$). That is, for any QPT algorithm $\mathcal{A}$ against the key-indistinguishability game (defined in Figure 5.2),* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$ *is negligible under the* dual-prf*,* prf*,* ind-cca*,* ind-cca *and* sufcma *security of the* PRF*,* PRF*,* KEM*,* KEM *and* MAC *primitives respectively. Thus we have :* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cca},\mathcal{A}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{MAC},n_P,n_S}(\lambda)$.

Proof. We now turn to proving our result. We split our analysis into three mutually-exclusive cases:

1. **Case 1** assumes that the test session π^s_i (such that $\mathcal{A}$ issued $\mathsf{Test}(i,s)$) is an initiator session, and that π^s_i has no *matching partner* (as in Figure 5.3). We define the QPT algorithm $\mathcal{A}$'s advantage in **Case 1** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$.

2. **Case 2** assumes that the test session π^s_i is a responder , and that π^s_i has no *matching partner*. We define $\mathcal{A}$'s advantage in **Case 2** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$.

3. **Case 3** assumes that the test session π^s_i has a *matching partner*. We define $\mathcal{A}$'s advantage in **Case 3** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$.

It is clear that: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$, thus we bound $\mathcal{A}$'s advantage in each case separately.

In **Case 1** and **Case 2** we show that $\mathcal{A}$'s advantage in causing the test session π_i^s to accept without a matching partner is negligible, and thus the $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible (since the experiment does not differ based on the challenge bit b, as the π_i^s does not compute a real-or-random session key).

In **Case 3** we replace the computation of the real session key by the test session π_i^s with a uniformly random key. Thus, the distribution of the keys returned by π_i^s are identical, regardless of the value of the challenge bit b, and we can show that $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible. We now begin with the first case.

Case 1: Test `init` session without origin session

We begin by showing that $\mathcal{A}$ has negligible advantage in causing π_i^s to reach an `accept` state without a matching session. We do so via the following sequence of game hops:

Game 0 This is the HAKE security game and $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) = \Pr(break_0)$.

Game 1 In this game, we guess the index (i,s) and the intended partner j and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi_{i'}^{s'}$ such that $\pi_{i'}^{s'}.pid = j'$ and $(i,s,j) \neq (i',s',j')$. Thus: $\Pr(break_0) \leq {n_P}^2 \cdot n_S \cdot \Pr(break_1)$.

Game 2 In this game we abort if the test session π_i^s sets the status $\pi_i^s.\alpha \leftarrow$ `reject`. Note that by the previous game we abort if the Test query is issued to a session that is not π_i^s, and if $\pi_i^s.pid \neq j$. If the session π_i^s ever reaches the status $\pi_i^s.\alpha \leftarrow$ `reject`, then the challenger will respond to the $\mathsf{Test}(i,s)$ query with $\perp$, and thus the difference in $\mathcal{A}$'s advantage between **Game 2** and **Game 3** is 0. Thus: $\Pr(break_1) \leq \Pr(break_2)$.

Game 3 In this game we define an abort event $\mathbf{abort}_\alpha$ that triggers if the test session π_i^s sets the status $\pi_i^s.\alpha \leftarrow$ `accept`. We note that the response to $\mathsf{Test}(i,s)$ issued by $\mathcal{A}$ is always $\perp$, (since the challenger aborts the game if π_i^s accepts, and $\mathsf{Test}(i,s) = \perp$ when π_i^s rejects the protocol execution), and thus $\Pr(break_3) = 0$. In **Game 4** and **Game 5** we prove that the probability of $\mathcal{A}$ in causing $\mathbf{abort}_\alpha$ to trigger is negligible. Thus: $\Pr(break_2) \leq \Pr(\mathbf{abort}_\alpha)$.

Game 4 In this game, we replace the key pqk_{G} derived in the test session π_i^s with the uniformly random and independent value $\widetilde{pqk_{\mathsf{G}}}$. We define a reduction $\mathcal{B}_1$ that inter-

acts with a post-quantum ind-cca KEM challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$ (as described in Definition 20), and replaces G's long-term KEM public key $pqpk_{\mathsf{G}}$ with the public key output from $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. Note that $\mathcal{B}_1$ can do so at the beginning of the game, by guessing the identity of the partner session of Test in **Game 2**, A knows $pqpk_{\mathsf{G}}$ prior to protocol execution. When π_i^s should compute $pqctxt_{\mathsf{G}}$, $\mathcal{B}_1$ instead replaces the computation of $pqctxt_{\mathsf{G}}$ and pqk_{G} with the outputs of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, $\widetilde{pqctxt}$ and $\widetilde{pqk}$ respectively (and similarly for the (potential) partner session π_j^t). Whenever party j requires the use of the secret key to decapsulate a ciphertext $pqctxt' \neq \widetilde{pqctxt}$, $\mathcal{B}_1$ simply queries $pqctxt'$ to $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces the computation of pqk' with the output of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. Detecting the replacement of pqk_{G} with a uniformly random value $\widetilde{pqk}$ implies an efficient distinguishing QPT algorithm $\mathcal{B}_1$ against the post-quantum ind-cca security of KEM. Thus: $\Pr(\mathbf{abort}_\alpha) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \Pr(break_4)$.

Game 5-9 For brevity, we consolidate many PRF replacement hops into a single game. In this game we replace the computation of intermediate ms values $ms_1 = \mathsf{PRF}(ms_0, \widetilde{pqk})$ (where $ms_0 = \mathsf{HKDF.Extract}(ck_{\mathsf{E}}, \epsilon)$), $ms_2 = \mathsf{PRF}(\widetilde{ms_1}, ck_{\mathsf{A}})$, $ms_3 = \mathsf{PRF}(\widetilde{ms_2}, pqk_{\mathsf{A}})$, $ms_4 = \mathsf{PRF}(\widetilde{ms_3}, ck_{\mathsf{E}})$, $ms_5 = \mathsf{PRF}(\widetilde{ms_4}, pqk_{\mathsf{E}})$ with a uniformly random and independent values $\widetilde{ms_1}, \widetilde{ms_2}, \widetilde{ms_3}, \widetilde{ms_4}, \widetilde{ms_5} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define reductions $\mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6$ that initializes a dual-prf/prf challenger $\mathcal{C}_{\mathsf{dual\text{-}prf/prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ms_a}, k_X)$, where $a \in \{1:5\}$, and instead queries k_X to $\mathcal{C}_{\mathsf{dual\text{-}prf/prf}}$. $\mathcal{B}_b$, where $b \in \{2:6\}$, uses the output of the query $\widetilde{ms_a}$ to replace the computation of ms_a. Since $\widetilde{ms_{a-1}}$ is uniformly random and independent by **Game 4**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseQK}(i,s)$ or $\mathsf{CompromiseQK}(j,t)$, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf/prf}}$ is 0, then $\widetilde{ms_a} = \mathsf{PRF}(\widetilde{ms_{a-1}}, k_X)$ and we are in **Game 4**. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf/prf}}$ is 1, then $\widetilde{ms_a} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 4 + i**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_b$ against the dual-prf/prf security of PRF, and we find: $\Pr(break_4) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_3,\mathcal{B}_4,\mathcal{B}_5,\mathcal{B}_6}(\lambda) + \Pr(break_9)$.

Game 10 In this game we replace the computation of the MAC and session key $mk, k = \mathsf{HKDF.Expand}(\widetilde{ms_5}, \mathsf{Hash}(m_0 \| m_1), \text{“PQAGKEM”})$ with a uniformly random and independent value $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We do so by initializing a prf challenger and

querying the hash value $\mathsf{Hash}(m_0\|m_1)$, and use the output $\widetilde{mk}, \widetilde{k}$ from the prf challenger to replace the computation of mk, k. Since $\widetilde{ms_5}$ is uniformly random and independent by **Game 9**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseQK}(i,s)$ or $\mathsf{CompromiseQK}(j,t)$, this is a sound replacement. If the test bit sampled by the prf challenger is 0, then $\widetilde{mk}, \widetilde{k} = \mathsf{HKDF.Expand}(\widetilde{ms_5}, \mathsf{Hash}(m_0\|m_1), \text{“}\texttt{PQAGKEM}''\text{)}$ and we are in **Game 9**. If the test bit sampled by the prf challenger is 1, then $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 10**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful QPT adversary $\mathcal{B}_7$ against the post-quantum prf security of PRF, and we find: $\Pr(break_9) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_7}(\lambda) + \Pr(break_{10})$.

Game 11 In this game we abort if the test session π_i^s receives a message τ (computed over $m_0\|m_1$) that verifies correctly but there exists no honest session π_j^t that has output τ. Specifically, in **Game 11** we define a reduction $\mathcal{B}_8$ against the sufcma security of the Message Authentication Code MAC. When $\mathcal{B}_8$ needs to compute a MAC over $m_0\|m_1$ using $\widetilde{mk}$, $\mathcal{B}_8$ computes the MAC by initializing a sufcma challenger $\mathcal{C}_{\mathsf{sufcma}}$ and querying $m_0\|m_1$. No changes to the experiment occur, as $\mathcal{C}_{\mathsf{sufcma}}$ computes MACs identically to $\mathcal{C}$, so $\mathcal{A}$ cannot detect this replacement. In addition, $\widetilde{mk}$ is a uniformly random and independent value, by **Game 10**.

By the definition of **Case 1**, π_i^s sets the status $\pi_i^s.\alpha \leftarrow \texttt{accept}$ despite there being no honest session that matches π_i^s. Thus, $\mathcal{B}_8$ never queried $m_0\|m_1$ to $\mathcal{C}_{\mathsf{sufcma}}$, and it follows that τ is a forged message. Thus, if π_i^s receives a MAC tag τ (over $m_0\|m_1$) that verifies correctly but there exists no honest session π_j^t that matches π_i^s, then $(m_0\|m_1, \tau)$ represents a valid forgery and $\mathcal{B}_8$ wins the sufcma security game against MAC, and $\Pr(break_{10}) = \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_8}(\lambda)$.

Since π_i^s now aborts when verifying τ, it cannot trigger $\mathbf{abort}_\alpha$ and thus we have: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq {n_P}^2 n_S \cdot \big(\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_2}(\lambda) + 5 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_3,\mathcal{B}_4,\mathcal{B}_5,\mathcal{B}_6,\mathcal{B}_7}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_8}(\lambda) + \big)$.

Case 2: Test responder session without origin session

We now show that $\mathcal{A}$ has negligible change in causing π_i^s (with $\pi_i^s.\rho = \mathsf{responder}$) to reach an `accept` state without an origin session. As the proof of **Case 2** follows analogously to **Case 1** with minor changes (notation to account for changes in role, and minor changes in **Game 11** to account for authenticating $m_0\|m_1\|\tau$) instead of $m_0\|m_1$), we omit these game hops and proceed to **Case 3**.

Case 3: Test session with matching session

In **Case 3**, we show that if $\mathcal{A}$ that has issued a $\mathsf{Test}(i,s)$ query to a clean session π_i^s, then $\mathcal{A}$ has negligible advantage in guessing the test bit b. In what follows, we note that for the cleanness predicate $\mathsf{clean}_{q\mathsf{HAKE}}$ to be upheld by π_i^s, then $\mathsf{CompromiseQK}(i,s)$, $\mathsf{CompromiseQK}(j,t)$ cannot be queried (where π_j^t *matches* π_i^s). Thus, we can assume in what follows that $\mathcal{A}$ has not compromised the *post-quantum ephemeral* KEM secrets. We now show that $\mathcal{A}$ has negligible advantage in guessing the test bit b, via the following series of game hops:

Game 0 This is the HAKE security game, and $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) = \Pr(break_0)$.

Game 1 In this game, we guess the index (i,s) and the matching session (j,t) and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi_{i'}^{s'}$ such that $\pi_{t'}^{j'}$ matches $\pi_{i'}^{s'}$ and $(i,s),(j,t) \neq (i',s'),(j',t')$. Thus $\Pr(break_0) \leq n_P^2 \cdot n_S^2 \cdot \Pr(break_1)$.

Game 2 In this game, we replace the key pqk_{E} derived in the test session π_i^s with the uniformly random and independent value $\widetilde{pqk}$. We do so by defining a reduction $\mathcal{B}_9$ who interacts with a post-quantum ind-cca KEM challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$ (as described in Definition 23) and replaces the $pqpk_{\mathsf{E}}$ value sent in m_0 with the public-key $\widetilde{pqpk}$ received from $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. When potential partner should compute the ciphertext $pqctxt$ sent in m_1, $\mathcal{B}_9$ instead replaces $pqctxt$ and pqk_{E} with $\widetilde{pqctxt}$ and $\widetilde{pqk}$ respectively. Whenever party j requires the use of the secret key to decapsulate a ciphertext $pqctxt' \neq \widetilde{pqctxt}$, $\mathcal{B}_9$ simply queries $pqctxt'$ to $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces the computation of pqk' with the output of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. Since π_i^s matches π_j^t, we know that the public-key and ciphertext sent in m_0 and m_1 respectively were received by the sessions without modification. Detecting the replacement of pqk_{E} with a uniformly random value $\widetilde{pqk}$ implies an efficient distinguishing QPT algorithm $\mathcal{B}_9$ against the post-quantum ind-cca security of KEM. Thus: $\Pr(break_1) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_9}(\lambda) + \Pr(break_2)$.

Game 3 In this game we replace the computation of intermediate ms $ms_5 = \mathsf{PRF}(ms_4, \widetilde{pqk})$ with a uniformly random and independent value $\widetilde{ms_5} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_{10}$ that initializes a dual-prf challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ when π_i^s needs to compute $\mathsf{PRF}(ms_4, \widetilde{pqk})$ and instead queries ms_4 to $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. $\mathcal{B}_{10}$ uses the output of the query $\widetilde{ms_5}$ to replace the computation of ms_5. Since $\widetilde{pqk}$ is uniformly random and independent by

Game 2, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseQK}(i,s)$ or $\mathsf{CompromiseQK}(j,t)$, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 0, then $\widetilde{ms_5} = \mathsf{PRF}(ms_4, \widetilde{pqk})$ and we are in **Game 2**. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 1, then $\widetilde{ms_5} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 3**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_{10}$ against the dual-prf security of PRF, and we find: $\Pr(break_2) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{10}}(\lambda) + \Pr(break_3)$.

Game 4 In this game we replace the computation of the MAC and session key $mk, k = \mathsf{HKDF.Expand}(\widetilde{ms_5}, \mathsf{Hash}(m_0\|m_1), \text{“}\texttt{PQAGKEM}''\text{)}$ with a uniformly random and independent value $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We do so by initializing a prf challenger and querying the hash value $\mathsf{Hash}(m_0\|m_1)$, and use the output $\widetilde{mk}, \widetilde{k}$ from the prf challenger to replace the computation of mk, k. Since $\widetilde{ms_5}$ is uniformly random and independent by **Game 3**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseQK}(i,s)$ or $\mathsf{CompromiseQK}(j,t)$, this is a sound replacement. If the test bit sampled by the prf challenger is 0, then $\widetilde{mk}, \widetilde{k} = \mathsf{HKDF.Expand}(\widetilde{ms_5}, \mathsf{Hash}(m_0\|m_1), \text{“}\texttt{PQAGKEM}''\text{)}$ and we are in **Game 3**. If the test bit sampled by the prf challenger is 1, then $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 4**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful QPT adversary $\mathcal{B}_{11}$ against the post-quantum prf security of PRF, and we find: $\Pr(break_3) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{11}}(\lambda) + \Pr(break_4)$.

Since $\widetilde{k}$ is now uniformly random and independent value of the protocol flow regardless of the value of the test bit b, $\mathcal{A}$ has no advantage in guessing the test bit and thus:
$\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{q\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq {n_P}^2 {n_S}^2 \cdot \big(\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_9}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{10}}(\lambda) + \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{11}}(\lambda)\big)$.

□

Next we present the security of PQAG-KEM against purely classical adversaries, i.e. $\mathcal{A}$ is a probabilistic polynomial-time algorithm, in Theorem 11.

Theorem 11 (PQAG-KEM Classical Security)**.** *The* PQAG-KEM *protocol presented in Section 6.2 is secure under cleanness predicate* $\mathsf{clean}_{c\mathsf{HAKE}}$ *(capturing perfect forward security and resilience to KCI attacks against classical adversaries). That is, for any PPT algorithm* $\mathcal{A}$ *against the key-indistinguishability game (defined in Figure 5.2),* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$ *is negligible under the* dual-prf*,* prf*,* ind-cca*,* ind-cca *and* sufcma *security of the* PRF*,* PRF*,* KEM*,* KEM*, and* MAC *primitives respectively. Thus we have :* $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{ind\text{-}cca},\mathcal{A}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{MAC},n_P,n_S}(\lambda)$.

Proof. We now turn to proving our result. We split our analysis into three mutually-exclusive cases:

1. **Case 1** assumes that the test session π_i^s (such that $\mathcal{A}$ issued $\mathsf{Test}(i,s)$) is an $\mathsf{initiator}$ session, and that π_i^s has no *matching partner* (as in Figure 5.3). We define the PPT algorithm $\mathcal{A}$'s advantage in **Case 1** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$.

2. **Case 2** assumes that the test session π_i^s is a $\mathsf{responder}$, and that π_i^s has no *matching partner*. We define $\mathcal{A}$'s advantage in **Case 2** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$.

3. **Case 3** assumes that the test session π_i^s has a *matching partner*. We define the PPT algorithm $\mathcal{A}$'s advantage in **Case 3** as $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$.

It is clear that: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{c\mathsf{HAKE}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda)$, thus we bound $\mathcal{A}$'s advantage in each case separately.

In **Case 1** and **Case 2** we show that $\mathcal{A}$'s advantage in causing the test session π_i^s to accept without a matching partner is negligible, and thus the $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible (since the experiment does not differ based on the challenge bit b, as the π_i^s does not compute a real-or-random session key). As the analysis of **Case 1** and **Case 2** follows identically the corresponding analysis of Theorem 10, we omit the details here, and point the reader to the formal proofs for Theorem 10.

In **Case 3** we replace the computation of the real session key by the test session π_i^s with a uniformly random key. Thus, the distribution of the keys returned by π_i^s are identical, regardless of the value of the challenge bit b, and we can show that $\mathcal{A}$'s advantage in winning the key-indistinguishability game is negligible. We now begin with the third case.

Case 3: Test session with matching session

In **Case 3**, we show that if $\mathcal{A}$ that has issued a $\mathsf{Test}(i,s)$ query to a clean session π_i^s, then $\mathcal{A}$ has negligible advantage in guessing the test bit b. In what follows, we split our analysis of Case 3 into the following sub-cases, each corresponding to a condition necessary for the cleanness predicate $\mathsf{clean}_{c\mathsf{HAKE}}$ to be upheld by π_i^s. These are the subcases (where π_j^t *matches* π_i^s):

– $\mathsf{CompromiseQK}(i,s)$, $\mathsf{CompromiseQK}(j,t)$ were not queried.

– $\mathsf{CompromiseCK}(i,s)$, $\mathsf{CompromiseCK}(j,t)$ were not queried.

It is straightforward to see that the advantage of $\mathcal{A}$ in Case 3 is bound by the sum of the advantages of $\mathcal{A}$ in all subcases. It is also straightforward to see that the proof of **Subcase 3.1** follows identically the corresponding analysis of **Case 3** of Theorem 10, and so we omit the details here, and point the reader to the formal proofs for Theorem 10. We now treat the second subcase (**Subcase 3.2**), where $\mathcal{A}$ has not compromised the *classic ephemeral* KEM secrets.

3.2: $\mathsf{CompromiseCK}(i,s)$, $\mathsf{CompromiseCK}(j,t)$ have not been issued, where π_j^t *matches* π_i^s.

Game 0 This is the HAKE security game, and $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{cHAKE},\mathcal{A},\mathbf{C3.2}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) = \Pr(break_0)$.

Game 1 In this game, we guess the index (i,s) and the matching session (j,t) and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi_{i'}^{s'}$ such that $\pi_{t'}^{j'}$ matches $\pi_{i'}^{s'}$ and $(i,s),(j,t) \neq (i',s'),(j',t')$. Thus $\Pr(break_0) \leq n_P^2 \cdot n_S^2 \cdot \Pr(break_1)$.

Game 2 In this game, we replace the key ck_{E} derived in the test session π_i^s with the uniformly random and independent value $\widetilde{ck}$. We define a reduction $\mathcal{B}_{12}$ that interacts with a classical $\mathsf{ind\text{-}cca}$ KEM challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$ and replace the cpk_{E} value sent in m_0 with the public key $\widetilde{cpk}$ received from $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. When potential partner should compute ciphertext $cctxt$ sent in m_1, $\mathcal{B}_{12}$ instead replaces the computation of $cctxt$ and ck_{E} with the outputs of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, $\widetilde{cctxt}$ and $\widetilde{ck}$ respectively. Whenever party j requires the use of the secret key to decapsulate a ciphertext $cctxt' \neq \widetilde{cctxt}$, $\mathcal{B}_{12}$ simply queries $cctxt'$ to $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces the computation of ck' with the output of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. Since π_i^s matches π_j^t, we know that the public-key and ciphertext sent in m_0 and m_1 respectively were received by the sessions without modification. Detecting the replacement of ck_{E} with a uniformly random value $\widetilde{ck}$ implies an efficient distinguishing PPT algorithm $\mathcal{B}_{12}$ against the classical $\mathsf{ind\text{-}cca}$ security of KEM. Thus: $\Pr(break_1) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_{12}}(\lambda) + \Pr(break_2)$.

Game 3 In this game we replace the computation of intermediate ms $ms_4 = \mathsf{PRF}(ms_3, \widetilde{ck})$ with a uniformly random and independent value $\widetilde{ms_4} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_{13}$ that initializes a $\mathsf{dual\text{-}prf}$ challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ when π_i^s needs to compute $\mathsf{PRF}(ms_3, \widetilde{ck})$

and instead queries ms_3 to $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. $\mathcal{B}_{13}$ uses the output of the query $\widetilde{ms_4}$ to replace the computation of ms_4. Since $\widetilde{ck}$ is uniformly random and independent by **Game 2**, and $\mathcal{A}$ cannot issue $\mathsf{CompromiseCK}(i,s)$ or $\mathsf{CompromiseCK}(j,t)$, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 0, then $\widetilde{ms_4} = \mathsf{PRF}(ms_3, \widetilde{ck})$ and we are in **Game 2**. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 1, then $\widetilde{ms_4} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 3**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_{13}$ against the $\mathsf{dual\text{-}prf}$ security of PRF, and we find: $\Pr(break_2) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{13}}(\lambda) + \Pr(break_3)$.

Game 4 In this game we replace the computation of intermediate ms $ms_5 = \mathsf{PRF}(\widetilde{ms_4}, pqk_{\mathsf{E}})$ with a uniformly random and independent value $\widetilde{ms_5} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction $\mathcal{B}_{14}$ that initializes a prf challenger $\mathcal{C}_{\mathsf{prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ms_4}, pqk_{\mathsf{E}})$ and instead queries pqk_{E} to $\mathcal{C}_{\mathsf{prf}}$. $\mathcal{B}_{14}$ uses the output of the query $\widetilde{ms_5}$ to replace the computation of ms_5. Since $\widetilde{ms_4}$ is uniformly random and independent by **Game 3**, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 0, then $\widetilde{ms_5} = \mathsf{PRF}(\widetilde{ms_4}, pqk_{\mathsf{E}})$ and we are in **Game 3**. If the test bit sampled by $\mathcal{C}_{\mathsf{prf}}$ is 1, then $\widetilde{ms_5} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 4**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_{14}$ against the prf security of PRF, and we find: $\Pr(break_3) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{14}}(\lambda) + \Pr(break_4)$.

Game 5 In this game we replace the computation of the MAC and session key $mk, k = \mathsf{HKDF.Expand}(\widetilde{ms_5}, \mathsf{Hash}(m_0 \| m_1), \text{“}\texttt{PQAGKEM}''\text{)}$ with a uniformly random and independent value $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We do so by initializing a prf challenger and querying the hash value $\mathsf{Hash}(m_0 \| m_1)$, and use the output $\widetilde{mk}, \widetilde{k}$ from the prf challenger to replace the computation of mk, k. Since $\widetilde{ms_5}$ is uniformly random and independent by **Game 4**, this is a sound replacement. If the test bit sampled by the prf challenger is 0, then $\widetilde{mk}, \widetilde{k} = \mathsf{HKDF.Expand}(\widetilde{ms_5}, \mathsf{Hash}(m_0 \| m_1), \text{“}\texttt{PQAGKEM}''\text{)}$ and we are in **Game 4**. If the test bit sampled by the prf challenger is 1, then $\widetilde{mk}, \widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 5**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful PPT adversary $\mathcal{B}_{15}$ against the post-quantum prf security of PRF, and we find: $\Pr(break_4) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{15}}(\lambda) + \Pr(break_5)$.

Since $\widetilde{k}$ is now uniformly random and independent value of the protocol flow regardless

of the value of the test bit b, $\mathcal{A}$ has no advantage in guessing the test bit and thus: $\mathsf{Adv}^{\mathsf{HAKE},\mathsf{clean}_{\mathsf{cHAKE}},\mathcal{A},\mathbf{C3.2}}_{\mathsf{PQAG\text{-}KEM},n_P,n_S}(\lambda) \leq {n_P}^2 {n_S}^2 \cdot \big(\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_{12}}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{13}}(\lambda) + 2 \cdot \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{B}_{14},\mathcal{B}_{15}}(\lambda)\big)$.

□

6.5 Conclusion

While the aviation industry is traditionally cautious about rapid technological change, the increasing sophistication of cyber threats necessitates a proactive approach to securing CPDLC communication. Failure to do so could compromise the integrity and safety of critical aviation systems, potentially leading to severe consequences. This chapter introduced two novel hybrid key exchange protocols, PQAG-SIG and PQAG-KEM, designed to improve the insecure CPDLC communication between ground stations and aircraft in the face of evolving adversarial capabilities. We rigorously proved the security of these protocols against both quantum and classical adversaries. By instantiating our protocols with various post-quantum algorithms, we demonstrated their practical applicability to the resource-constrained avionic environment. Comparative analysis with existing solutions revealed that our proposals offer superior performance and robust security.

Chapter 7

Post Quantum Handover Protocol for Aviation

In this chapter we propose a post-quantum secure handover construction. Our PQAG-HO scheme facilitates a secure CPDLC handover of an aircraft (A) transitioning from one ground stations (G_1) to another (G_2). By incorporating our KEX and HO formalisms within the design of PQAG-HO, we reduce the computational overhead of complex cryptographic operations. We provide a formal proof of security for the proposed protocol against a quantum adversary. We further instantiate our proposed protocol with post-quantum Kyber algorithm to understand the real-world applicability of our protocol within the resource-constrained ecosystem of avionic communication. We begin this chapter with a discussion on the current state-of-the-art in avionic handovers.

7.1 Avionic Handover Schemes

As an aircraft travels from one geographic location to another, the Controller-Pilot Data Link Communications (CPDLC) protocol facilitates an automatic transference of its current communication session, eliminating the requirement to re-establish a new session every time an aircraft enters a subsequent geographic zone [65]. At present all CPDLC communications are carried out over unencrypted and unauthenticated datalinks. Figure 7.1 illustrates an expected implementation of a CPDLC connection handover as described by official ICAO guidelines [65]. Thus, unsurprisingly, CPDLC does not provide any security guarantees,

since all communications are carried out over unencrypted and unauthenticated channels. Smailes et al. [115] explore practical attacks against CPDLC, focusing on impersonating ATC stations to aircrafts. They successfully hijack a session between an aircraft and a legitimate ATC station during the handover phase. In their attack they trigger a false handover to a new ATC station by injecting messages into an existing CPDLC session. Their attacks may have catastrophic consequences, as CPDLC messages contain critical instructions as declaring emergencies, changing altitudes or changing speeds of an aircraft, which Smailes et al. [115] show can easily be hijacked and tampered with by an unsophisticated attacker. They highlight the practicality of these attack by launching them as an attacker stationed hundreds of kilometers away from their targets.

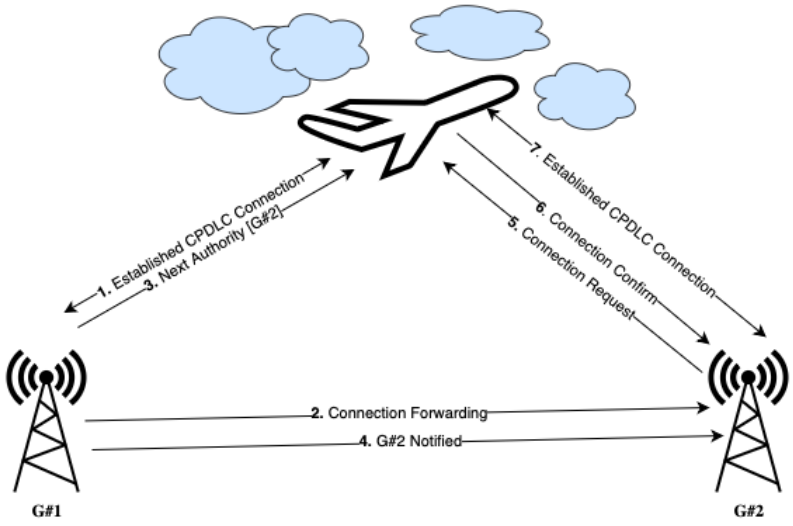


Figure 7.1: A generic insecure CPDLC handover between source ground station G#1*, target ground station* G#2 *and aircraft* A.

The work of Maurer et al. [87] proposes a secure handover scheme for aviation that provides the flexibility of using either classical or post-quantum primitives within their construction. In their scheme, the source ground station S acts as an intermediary throughout the handover, forwarding messages back and forth between an aircraft A and the target ground station T until the session handover is completed. They formally verify the forward secrecy and mutual authentication properties of the proposed scheme using the Tamarin symbolic verifier. They extend their work to cover multiple aviation handover scenarios in

[72] and incorporate physical unclonable functions (PUF) and post quantum BIKE primitive within their constructions. However, the purported security properties are only informally analyzed and no formal proofs are discussed. Compared to these existing solutions, our PQAG-HO offers a more efficient and secure approach. By streamlining the cryptographic operations, we reduce computational overhead without sacrificing formal security guarantees. We further validate our claims by instantiating, implementing, and evaluating our construction.

7.2 PQAG-HO Handover Protocol

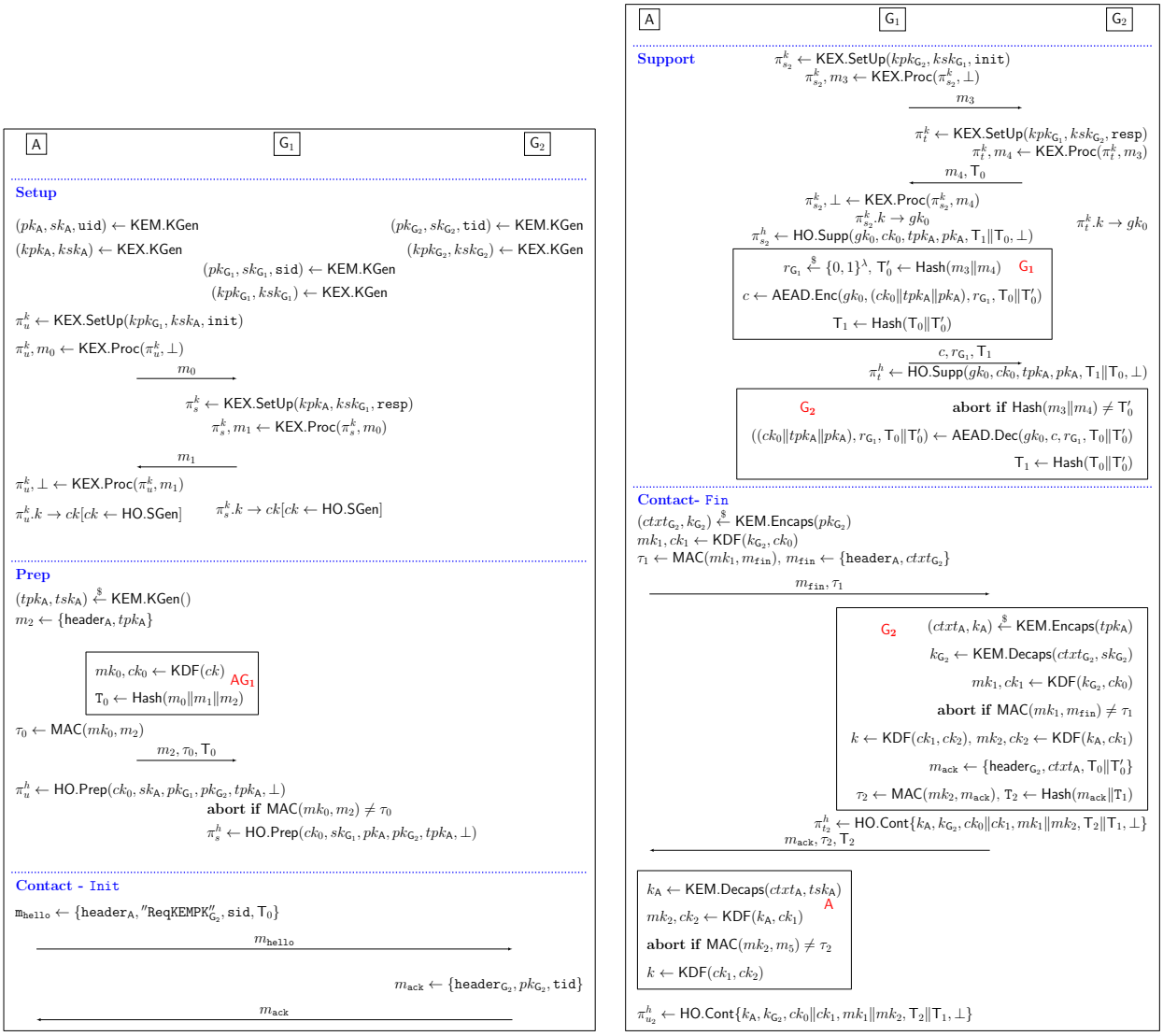
In this section we describe our proposed post-quantum air-to-ground handover protocol PQAG-HO, executed between an Aircraft A and two Ground Stations G_1 and G_2, as A geographically leaves the area covered by G_1 to enter the area covered by G_2. Our construction illustrated in Figure 7.2 utilize and realizes in practice our handover framework discussed in Chapter 3. Particularly, PQAG-HO delves into the underlying cryptographic operations that take place during the Setup stage of its construction unlike our StrongHO construction presented in Section 3.3.

StrongHO, also modeled within our handover formalism, presents an instantiation of a strong HO construction but abstracts away the specifics of its Setup phase. Moreover, while StrongHO provides the strongest security guarantees within our HO framework, its practical deployment in resource-constrained aviation environments is hindered by its significant computational overhead. Specifically, its reliance on computationally expensive operations, such as public-key encryption and puncturable PRFs, poses challenges for real-time implementation. In contrast, PQAG-HO is tailored specifically to the aviation domain, focusing only on the security properties relevant to that context. Our PQAG-HO design is optimized to reduce computational overhead. We limit expensive cryptographic operations, such as key exchange, to initial key negotiations between participating entities within our KEX formalism and reduce the computational overhead on aircraft A during the Cont phase of PQAG-HO. We observe due to the nature of its design, PQAG-HO enables seamless and secure transition of the existing session between A and G_1 to A and G_2 without the need to perform any additional KEX key exchange negotiations. We note that unlike StrongHO, PQAG-HO provides a more detailed and fine-grained view of the cryptographic operations within the Setup phase. We observe that PQAG-HO aptly showcases the practical application of our theoretical frameworks, the HO handover formalism (Chapter 3) and the KEX

secure key exchange formalism (Chapter 5). By seamlessly incorporating both KEX and HO into its design, PQAG-HO illustrates how these formalisms can be used to create secure and efficient protocol constructions, reinforcing the reciprocal nature of their relationship.

We present the detailed cryptographic operations that take place during PQAG-HO in Figure 7.2. During the initial Setup phase, A and G_1 executes an authenticated key exchange to derive an initial fresh shared key, which will be used for further key derivations in subsequent HO phases. We model this key exchange using our KEX formalization presented in Section 5.2. During the Supp phases of PQAG-HO, G_1 and G_2 independently run a separate KEX to derive a different shared key. We note that the KEX formalism abstracts away an underlying PQAG-KEM execution as described in Section 6.2.1. This additional KEX execution provides perfect forward secrecy for PQAG-HO by deriving a fresh shared key for the Supp phase. Below we explain PQAG-HO in more detail.

- **Setup:** During this phase, the protocol retrieves and utilizes existing identities and keys from prior PQAG-HO executions, if available. These include entity public key pairs and a fresh chain key ck established between A and G_1 through a KEX execution. In the absence of previous PQAG-HO executions, the protocol generates the necessary keys and identities according to the specific instantiation. In order to agree on ck, an initial key establishment KEX execution takes place. First, A communicates m_0 to G_1, which was output by a KEX.Proc and contains the relevant parameters from A to agree on a shared state with G_1. Upon reception, G_1 uses m_0 as input to its own KEX.Proc, which produces a shared state π and m_1 containing the complementary parameters from G_1 to agree on a shared state with A. G_1 returns m_1 to A, which A uses to agree on a shared state with G_1 and derive ck. We note that the chain key ck is analogous to the bootstrap key bk in StrongHO.
- **Preparation:** After agreeing on ck, during this phase A communicates a KEM public key to G_1 to be forwarded to G_2 (in an authenticated manner). A and G_1 both derive new keys mk_0, ck_0 from the chaining key ck. A then generates an ephemeral KEM key-pair $tpk_{\mathsf{A}}, tsk_{\mathsf{A}}$. This ensures the confidentiality of the session keys generated, i.e. achieving perfect forward secrecy for the subsequent session between A and G_2. After that, A creates a message m_2, which includes a header and tpk_{A}. Then, A sends m_2 and its MAC tag τ_0 to G_1. Lastly, both A and G_1 calculate T_0 which is a hash digest over all messages exchanged since the handover has initiated.
- **Support:** During this phase, similar to the KEX execution during Prep, G_1 and G_2



(a) PQAG-HO *Protocol Stage 1.*

(b) PQAG-HO *Protocol Stage 2.*

Figure 7.2: PQAG-HO *Protocol (Stages have been separated for readability).*

execute another KEX and securely derive session key gk_0, and G_1 securely forwards the keying material received from A in the previous phase to G_2. After deriving gk_0, G_1 generates nonce r_{G_1} and calculates the hash digest T'_0 over m_3 and m_4 ($m_3\|m_4$) exchanged during the KEX that produced gk_0. G_1 under gk_0 next encrypts keying material ($ck_0\|tpk_\mathsf{A}\|pk_\mathsf{A}$) using an authenticated encryption scheme AEAD. Lastly, G_1

calculates the hash digest T_1 over $(\mathsf{T}_0 \| \mathsf{T}'_0)$.

– **Contact:** During this phase, A authenticates G_2 and establishes a shared secret state. After learning the identity of G_2 via $\mathtt{tid}$ and pk_{G_2}, A starts the authentication process by encapsulating under pk_{G_2} and deriving a set of keys (mk_1, ck_1) with the encapsulated key k_{G_2} and ck_0 derived during Prep. A then creates $m_{\mathtt{fin}}$, which includes header and encapsulated ciphertext $ctxt_{\mathsf{G}_2}$, and sends it with its MAC tag τ_1 to G_2. When G_2 receives $m_{\mathtt{fin}}$, they decapsulate the ciphertext to obtain k_{A} and proceed to derive (mk_1, ck_1) and verify τ_1. Upon successful verification, G_2 proceeds to derive a set of keys (mk_2, ck_2) and a fresh session key k. G_2 the constructs $m_{\mathtt{ack}}$ which contains a header, encapsulation ciphertext $ctxt_{\mathsf{A}}$ of tpk_{A} and previous transcripts $(\mathsf{T}_0 \| \mathsf{T}'_0)$. G_2 sends $m_{\mathtt{ack}}$ to A along with its MAC tag τ_2 and hash digest T_2 calculated over $(m_{\mathtt{ack}} \| \mathsf{T}_1)$. Upon reception, A decapsulates the ciphertext to obtain k_{A} and computes (mk_2, ck_2, k). A further verifies τ_2 to check message integrity.

7.3 Execution Environment

The execution environment for $\mathsf{PQAG\text{-}HO}$ security framework is largely identical to our $\mathsf{StrongHO}$ scheme proposed in Chapter 3. Below we discuss the details of the execution environment for $\mathsf{PQAG\text{-}HO}$ in detail, which closely follows 3.2.2 with minor modifications.

Each session π_i^s of $\mathsf{PQAG\text{-}HO}$ the following set of per-session variables are maintained :

– $\rho \in \{\mathtt{U}, \mathtt{S}, \mathtt{T}\}$: The role of the party in the current session.
– $i \in \{1, ..., n_P\}$: Index of the session owner.
– $s \in \{1, ..., n_S\}$: Current session index.
– $t \in \{1, ..., n_S\}$: Previous session index.
– $T_{\mathtt{P}}, T_{\mathtt{S}}, T_{\mathtt{C}}$: Session transcripts of the Prep, Supp and Cont algorithms respectively, initialised by $\perp$.
– $\alpha \in \{\mathtt{prep}, \mathtt{supp}, \mathtt{hand}, \mathtt{accept}, \mathtt{reject}, \perp\}$: The current status of the session, initialised with $\perp$.
– $k \in \{\{0,1\}^\lambda, \perp\}$: Session key to be used in some following symmetric key protocol, or $\perp$ if no session key has yet been computed.
– $ck \in \{\{0,1\}^\lambda, \perp\}$: Chaining key used as the initial shared secret between the A and the G_1, or $\perp$ if no key has yet been computed. This key is derived during the Setup stage is the output of a KEX.

– $gk \in \{\{0,1\}^\lambda, \perp\}$: Shared key derived during the Supp stage between the G_1 and the G_2, or $\perp$ if no key has yet been computed. This key too is the output of a KEX.

Compared to StrongHO which captures multiple HO-specific notions of security, PQAG-HO predominantly captures the notion of *key indistinguishability* presented in Figure 3.3. This is due to the domain-specific requirements of commercial aviation, which require constant tracking of flights on air. As such, additional security properties as *unlinkability* and *path privacy* captured in StrongHO are not desirable within the domain of commercial aviation. In order to address the strength of a quantum capable adversary, we slightly modify the *key indistinguishability* definition for StrongHO, which we describe in Definition 44 under the cleanness predicated 26. While typically cleanness predicates are tailored to individual constructions, we observe that for PQAG-HO, the notion of key indistinguishability is achieved in a manner identical to our StrongHO which captures a perfect forward secret handover scheme. Therefore, we refer the reader to the StrongHO cleanness predicate $\mathsf{clean}_{\mathsf{str\text{-}kind}}$ in Definition 26 and describe our KIND-security definition for PQAG-HO as follows.

Definition 44 (KIND Key Indistinguishability)**.** *Let* HO *be a secure handover protocol, and* n_P, $n_S \in \mathbb{N}$. *For a particular given predicate* clean*, and a QPT algorithm* $\mathcal{A}$*, we define the advantage of* $\mathcal{A}$ *in the* KIND *key indistinguishability game to be:* $\mathsf{Adv}^{\mathsf{KIND,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{KIND,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$. *We say that* HO *is* KIND*-secure if, for all* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{KIND,clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

We formally prove the KIND security of Definition 44 within our PQAG-HO construction in Section 7.5.

7.4 PQAG-HO Implementation

In this section we discuss our instantiation and simple reference implementation of the PQAG-HO. We implement PQAG-HO in Python, and benchmarked its walltime performance on a standard desktop system. We begin by discussing our choices of instantiations for cryptographic primitives.

Instantiation

PQAG-HO aims for 128-bit post-quantum security against a quantum-equipped attacker. We instantiate PQAG-HO with Kyber as the KEM used for all KEM operations depicted in Figure 7.2. We note that all KEX executions within PQAG-HO have been instantiated with PQAG-KEM described in Section 6.2.1 and instantiated in Section 6.3. For all our KEX processes, we derive a final hybrid session key that combines outputs of classical CKEM (ECC-DH) and post-quantum PQKEM (Kyber). For our instantiation, the final derived shared key between A and G_2 was 256-bits long. Thus our choices of cryptographic algorithms for PQAG-HO are:

- Classic CKEM: Elliptic-curve DH key exchange using curve P384 [70].
- Post-Quantum KEM: Kyber-512 [122], achieving 128-bit quantum security.
- AEAD: AESGCM [11] using 256-bit keys.
- Hash function [101] using sha256.
- MAC: HMAC-SHA-256 [75] using 256-bit keys.
- KDF: HKDF-SHA-256 [74] using 256-bit keys.

Implementation

We require the use of the Python `cryptography` library [122] for implementing CKEM and KDF, and for implementing post-quantum primitives we utilize the Python `pqcrypto` [100] library.

7.4.1 PQAG-HO Performance

We now profile the performance of the underlying cryptographic functions in terms of average execution times (for 100 iterations per each phase of PQAG-HO). We evaluate all Setup,Prep, Supp and Cont stages of PQAG-HO as active HO phases. Table 7.1 summarizes our results in terms of average walltime per 100 executions for each phase tested along with average round-trip-time per handover completion. Our experiments were performed on an Intel Core i7-11370H 3.30GHz CPU with 16GB RAM, running Windows 11 Home.

Predictably, Supp phase was the most time consuming, given it involves a KEX process as well as multiple other cryptographic operations including an authenticated encryption AEAD. Cont stage performed faster in comparison to Supp, although it involved multiple

Stage	Avg.Walltime	Std. Deviation
Setup	0.0027	0.0014
Prep	0.0004	0.00008
Supp	0.024	0.007
Cont	0.010	0.0047
PQAG-HO-RTT	0.037	0.0139

Table 7.1: PQAG-HO *Performance Evaluation (in seconds)*

post-quantum Kyber KGen, Encaps, Decaps operations. This can be explained by the difference between our KEX and KEM instantiations: while our KEX produces a hybrid key that combines classical ECC-DH with post-qauntum Kyber, our KEM instantiation derives a purely post-quantum Kyber key. Consequently, owing to Kyber's fast performance times and fewer overall calculations, Cont phase performed much faster compared to Supp. The fast key generation time of Kyber is also evident by the performance of Prep phase. Overall, the PQAG-HO averaged well under a second for a single round-trip with a standard deviation under 0.02 seconds. We highlight that, to the best of our knowledge, there is no other comparable works that benchmark their avionic handover constructions and that ours is the first of its kind.

7.5 Security Analysis

In this section we analyze the security of our proposed PQAG-HO and protocol, presented in 7.2. We prove the KIND security of the PQAG-HO protocol, described in Figure 3.7. For consistency and easy of readability we map the roles within our PQAG-HO construction to the parties of our formalized HO framework as follows: $\mathsf{A} = \mathtt{U}$; $\mathsf{G}_1 = \mathtt{S}$ and; $\mathsf{G}_2 = \mathtt{T}$. From this point onward, we will exclusively refer to the parties of our PQAG-HO construction as their formalized role within our HO framework. We begin by presenting the security of PQAG-HO, in Theorem 12.

Theorem 12 (PQAG-HO Security)**.** *The* PQAG-HO *protocol presented in Section 7.2 is post-quantum secure under cleanness predicate 26* $\mathsf{clean}_{\mathsf{str\text{-}kind}}$ *(capturing perfect forward security and resilience to KCI attacks against $\mathcal{A}$). That is, for any QPT algorithm $\mathcal{A}$*

against the key-indistinguishability game (defined in Figure 5.2), $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* dual-prf*,* hake*,* ind-cca*,* auth*,* sufcma *and* coll *security of the* PRF*,* PRF*,* HAKE*,* KEM*,* AEAD*,* MAC *and* Hash *primitives respectively. Thus we have:* $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{dual\text{-}prf},\mathcal{A}}_{\mathsf{PRF},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{hake},\mathcal{A}}_{\mathsf{HAKE},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{ind\text{-}cca},\mathcal{A}}_{\mathsf{KEM},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{auth},\mathcal{A}}_{\mathsf{AEAD},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{sufcma},\mathcal{A}}_{\mathsf{MAC},n_P,n_S}(\lambda) + \mathsf{Adv}^{\mathsf{coll},\mathcal{A}}_{\mathsf{Hash},n_P,n_S}(\lambda)$.

Proof. We proceed via a sequence of games. We bound the difference in the adversary's advantage in each game with the underlying cryptographic assumptions until the adversary reaches a game where the advantage of that game equals 0, which shows that adversary $\mathcal{A}$ cannot win.

We begin by dividing the proof into three separate cases (and denote with $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S,C_i}(\lambda)$ the advantage of the QPT adversary in winning the key-indistinguishability game in Case i) where the query $\mathsf{Test}(i,s)$ has been issued:

- Test session does not UT-match another session, $\mathsf{Corrupt}(i)$ has not been issued for T's long-term key before $\pi^s_i.\alpha = \texttt{accept}$ and $\pi^s_i.\rho = \texttt{U}$.
- Test session does not UT-match another session, $\mathsf{Compromise}(i,s)$ nor $\mathsf{Compromise}(l,r)$ (such that $\pi^s_i.ck = \pi^r_l.ck$) have been issued before $\pi^s_i.\alpha = \texttt{accept}$, $\mathsf{Corrupt}(i)$ has not been issued for S's long-term key before $\pi^s_i.\alpha = \texttt{accept}$ and $\pi^s_i.\rho = \texttt{T}$.
- Test session UT-match another session, and $\mathsf{Compromise}(i,s)$ and $\mathsf{Compromise}(l,r)$ (such that $\pi^s_i.ck = \pi^r_l.ck$) were never issued.

It follows that $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathsf{KIND},\mathsf{clean},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S,C_1}(\lambda) + \mathsf{Adv}^{\mathsf{KIND},\mathsf{clean},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S,C_2}(\lambda) + \mathsf{Adv}^{\mathsf{KIND},\mathsf{clean},\mathcal{A}}_{\mathsf{PQAG\text{-}HO},n_P,n_S,C_3}(\lambda)$

We begin with Case 1.

Case 1: $\mathsf{Corrupt}(i)$ **has not been issued for** T**'s long-term key before** $\pi^s_i.\alpha = \texttt{accept}$ **and** $\pi^s_i.\rho = \texttt{U}$. By the definition of the case, we assume that the QPT adversary $\mathcal{A}$ has not been able to compromise the long-term public key pk_{G} of the T session. Test session does not UT-match another session.

Game 0 This is the KIND security game and $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A},\mathbf{C1}}_{\mathsf{PQAG\text{-}HO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 In this game, we guess the index of the test session (i,s) and the intended source partner j and target partner l and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi^{s'}_{i'}$ such that $\pi^{s'}_{i'}.tpid = j', \pi^{s'}_{i'}.spid = l'$ and $(i,s,j,l) \neq (i',s',j',l')$. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq {n_P}^3 \cdot n_S \cdot \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda)$.

Game 2 In this game we replace ck computed in the test session π_i^s and its honest contributive keyshare session partner with a random key $\widetilde{ck}$ from the same key distribution. We explicitly define a reduction $\mathcal{B}$ who interacts with a hake challenger $\mathcal{C}_{\mathsf{hake}}$ (as described in Definition 43) as follows: At the beginning of the game $\mathcal{B}$ initializes $\mathcal{C}_{\mathsf{hake}}$ who provides $\mathcal{B}$ with a list of keys $(pk_1, \ldots, pk_{n_P})$. $\mathcal{B}$ also maintains a mapping for these keys between its own parties and the parties of $\mathcal{C}_{\mathsf{hake}}$. Whenever, $\mathcal{B}$ receives $\mathsf{Create}(i, \rho, j, \ell, s, t)$ $\mathcal{B}$ performs a set of checks. First $\mathcal{B}$ verifies if both s and t are $\perp$, indicating a fresh HO session. Then for all fresh Create queries where $\rho \in \{\mathtt{U}, \mathtt{S}\}$, $\mathcal{B}$ queries $\mathcal{C}_{\mathsf{hake}}$ with $\mathsf{Create}(i, j, \rho)$ where $\rho \in \{\mathtt{init}, \mathtt{resp}\}$ for $\rho = \mathtt{U}$ and $\rho = \mathtt{S}$ respectively. $\mathcal{C}_{\mathsf{hake}}$ will return a session identifier s' to $\mathcal{B}$ who will maintain a CREATE table that records the mapping of outputs between its own Create queries and the outputs from $\mathcal{C}_{\mathsf{hake}}$ as (i, s, s'). Next, whenever $\mathcal{B}$ receives a $\mathsf{Send}(i, s, m)$ query to a $\mathtt{U}$ or $\mathtt{S}$ session, $\mathcal{B}$ first checks if $\pi_i^s.\alpha = \mathsf{Setup}$. If so, $\mathcal{B}$ next checks if there exists a matching entry for (i, s, s') in its CREATE table and if it does exist, will send a $\mathsf{Send}(i, s', m)$ query to $\mathcal{C}_{\mathsf{hake}}$. If $\mathcal{A}$ issues a $\mathsf{Corrupt}(i)$ query to $\mathcal{B}$, $\mathcal{B}$ will issue corresponding $\mathsf{CorruptCK}(i), \mathsf{CorruptQK}(i)$ queries to $\mathcal{C}_{\mathsf{hake}}$ who will return a pair of keys (psk', csk'). If $\mathcal{A}$ issues a $\mathsf{Compromise}(i, s)$ query to $\mathcal{B}$, for any session that is not Test, $\mathcal{B}$ will in turn issue a $\mathsf{Reveal}(i, s')$ query to $\mathcal{C}_{\mathsf{hake}}$ to obtain some chain key ck' (if ck' was derived from a KEX execution and not some previous handover HO). Whenever $\mathcal{B}$ needs to calculate ck in the Test session (and any $\mathtt{US}$-matching sessions that compute ck), $\mathcal{B}$ queries $\mathcal{C}_{\mathsf{hake}}$ with $\mathsf{Test}(i, s')$ who returns a uniformly random $\widetilde{ck}$. Whenever $\mathcal{B}$ needs to calculate ck in non-Test sessions, $\mathcal{B}$ queries $\mathcal{C}_{\mathsf{hake}}$ with $\mathsf{Reveal}(i, s')$ who returns the correctly computed key ck. When the test bit sampled by $\mathcal{C}_{\mathsf{hake}}$ is 0, then $ck \leftarrow \pi_u^k.k$ and we are in **Game 1**. If the test bit sampled by $\mathcal{C}_{\mathsf{hake}}$ is 1, then $ck \leftarrow \mathcal{K}$ and we are in **Game 2**. The only difference between the two games is in this game $\mathcal{C}_{\mathsf{hake}}$ is performing all HAKE calculations which is indistinguishable to the adversary $\mathcal{A}$ for all computations produced by the actual HO execution. Any adversary that can detect this change can be turned into an adversary against the hake security and thus: $\mathsf{Adv}_{G_1}^{\mathcal{A}}(\lambda) \leq \mathsf{Adv}_{\mathsf{HAKE},\mathcal{A}}^{\mathsf{hake},\mathcal{B}}(\lambda) + \mathsf{Adv}_{G_2}^{\mathcal{A}}(\lambda)$.

Game 3 In this game we replace the computation of the extracted keys $mk_0, ck_0 \xleftarrow{\$} \mathsf{KDF}(\widetilde{ck}, \epsilon)$ with a uniformly random and independent value $\widetilde{mk_0}, \widetilde{ck_0} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π_i^s, and (potentially) its matching session π_j^t. We define a reduction

$\mathcal{B}_1$ that initializes a dual-prf challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ when π_i^s needs to compute $\mathsf{PRF}(\widetilde{ck}, \epsilon)$ and instead queries $\widetilde{ck}$ to $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. $\mathcal{B}_1$ uses the output of the query $\widetilde{mk_0}, \widetilde{ck_0}$ to replace the computation of mk_0, ck_0. Since $\widetilde{ck}$ is uniformly random and independent by **Game 2**, this is a sound replacement. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 0, then $\widetilde{mk_0}, \widetilde{ck_0} \xleftarrow{\$} \mathsf{KDF}(\widetilde{ck}, \epsilon)$ and we are in **Game 2**. If the test bit sampled by $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ is 1, then $\widetilde{mk_0}, \widetilde{ck_0} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ and we are in **Game 3**. Thus any adversary $\mathcal{A}$ capable of distinguishing this change can be turned into a successful adversary $\mathcal{B}_1$ against the dual-prf security of PRF, and we find: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda)$.

Game 4 In this game we abort if the test session π_i^s receives a message τ_0 (computed over m_2) that verifies correctly but there exists no honest session π_j^t that has output τ_0. Specifically, in **Game 4** we define a reduction $\mathcal{B}_2$ against the sufcma security of the Message Authentication Code MAC. When $\mathcal{B}_2$ needs to compute a MAC over m_2 using $\widetilde{mk_0}$, $\mathcal{B}_2$ computes the MAC by initializing a sufcma challenger $\mathcal{C}_{\mathsf{sufcma}}$ and querying m_2. No changes to the experiment occur, as $\mathcal{C}_{\mathsf{sufcma}}$ computes MACs identically to $\mathcal{C}$, so $\mathcal{A}$ cannot detect this replacement. Also, as the result of **Game 3**, $\widetilde{mk_0}$ is a uniformly random and independent value, so this replacement is sound. If π_i^s receives a MAC tag τ_0 (over m_2) that verifies correctly but there exists no honest session π_j^t that matches π_i^s, then (m_2, τ_0) represents a valid forgery and $\mathcal{B}_2$ wins the sufcma security game against MAC, and $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_2}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda)$.

Game 5 In a manner similar to **Game 2**, this game replaces gk_0 computed in the test session π_i^s and its honest contributive keyshare session partner with a random key $\widetilde{gk_0}$ from the same key distribution. We do so by explicitly defining a reduction $\mathcal{B}_3$ who interacts with a hake challenger $\mathcal{C}_{\mathsf{hake}}$. For all fresh HO sessions, whenever they receive a Create query where $\rho \in \{\mathtt{S}, \mathtt{T}\}$, $\mathcal{B}_3$ queries $\mathcal{C}_{\mathsf{hake}}$ with $\mathsf{Create}(i, j, role)$ where $role \in \{\mathtt{init}, \mathtt{resp}\}$ for $\rho = \mathtt{S}$ and $\rho = \mathtt{T}$ respectively. $\mathcal{B}_3$ maintains a CREATE table that maps the outputs of its own Create queries against the outputs from $\mathcal{C}_{\mathsf{hake}}$. For all $\mathsf{Send}(i, s, m)$ queries, $\mathcal{B}_3$ checks if $\pi_i^s.\alpha = \mathsf{Supp}$, and if so proceed in a manner similar to **Game 2**. Whenever $\mathcal{B}_3$ needs to calculate gk_0 in the Test session (and any ST-matching sessions that compute gk_0), $\mathcal{B}_3$ queries $\mathcal{C}_{\mathsf{hake}}$ with $\mathsf{Test}(i, s')$ who returns a uniformly random $\widetilde{gk_0}$. When the test bit sampled by $\mathcal{C}_{\mathsf{hake}}$ is 0, then $gk_0 \leftarrow \pi_t^k.k$ and we are in **Game 4**. If the test bit sampled by $\mathcal{C}_{\mathsf{hake}}$ is 1, then $\widetilde{gk_0} \leftarrow \{0,1\}^*$ and we are in **Game 5**. Any adversary that can detect this change can be turned into an adversary against the hake security and thus: $\mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda) \leq \mathsf{Adv}^{\mathsf{hake}}_{\mathsf{HAKE},\mathcal{B}_3}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda)$.

Game 6 Here we abort if a hash collision occurs, by computing all hash values honestly and aborting if there exists two evaluations $(m, \mathsf{Hash}(m)), (m', \mathsf{Hash}(m'))$ such that $m \neq m'$ but $\mathsf{Hash}(m) = \mathsf{Hash}(m')$. The challenger $\mathcal{C}$ interacts with a Hash-collision challenger $\mathcal{C}_{\mathsf{coll}}$, outputting the collision if found. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash},\mathcal{C}_{\mathsf{coll}}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda)$.

Game 7 In this game, we invoke abort $\mathtt{event}_\alpha$ if the adversary $\mathcal{A}$ is able to produce a value $c \leftarrow \mathsf{AEAD.Enc}(\widetilde{gk_0}, (ck_0 \| tpk_{\mathsf{A}} \| pk_{\mathsf{A}}), r_{\mathsf{G}_1}, \mathsf{T}_0 \| \mathsf{T}'_0)$ that decrypts correctly using $\widetilde{gk_0}$. Specifically, we introduce a reduction $\mathcal{B}_4$ that initializes an AEAD-auth challenger $\mathcal{C}_{\mathsf{auth}}$. Whenever it is required to encrypt/decrypt using $\widetilde{gk_0}$, $\mathcal{B}_4$ instead queries $\mathcal{C}_{\mathsf{auth}}$'s respective encryption/decryption oracles. By **Game 5** we know that $\widetilde{gk_0}$ is a uniformly random and independent value, and thus this substitution of keys is undetectable. If $\mathcal{A}$ can provide a ciphertext c' that decrypts correctly, but was never output by $\mathcal{C}_{\mathsf{auth}}$, then it follows that $\mathcal{A}$ has forged a ciphertext c' breaks the auth security of the AEAD scheme as in Definition 14. Therefore, T now aborts if they complete the support phase without a ST-matching partner, and $\Pr(\mathtt{event}_\alpha) = 0$. Thus, $\mathcal{A}$ cannot know any information about ck_0 before the protocol execution ends, and since gk_0 is secure by **Game 5**, nor is any information about ck_0 sent in the protocol transcript. Moreover, the additional-data field of c contains the concatenated Hash $\mathsf{T}_0 \| \mathsf{T}'_0$. By **Game 6** we abort the experiment if $\mathcal{A}$ causes a hash-collision to occur and any $\mathcal{A}$ that can trigger the abort event can be used by the challenger to break the underlying AEAD-auth security assumption. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) \leq \mathsf{Adv}^{\mathsf{auth}}_{\mathsf{AEAD},\mathcal{B}_4}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda)$.

Game 8 Similar to **Game 3**, this game replaces the computation of the extracted keys $mk_1, ck_1 \xleftarrow{\$} \mathsf{KDF}(k_{\mathsf{G}_2}, \widetilde{ck_0})$ with a uniformly random and independent value $\widetilde{mk_1}, \widetilde{ck_1} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ (where $\{0,1\}^{\mathsf{PRF}}$ is the output space of the PRF) used in the protocol execution of the test session π^s_i, and (potentially) its matching session π^t_j. We do this by defining a reduction $\mathcal{B}_5$ that initializes a dual-prf challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$ when π^s_i needs to compute $\mathsf{PRF}(k_{\mathsf{G}_2}, \widetilde{ck_0})$ and instead queries k_{G_2} to $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_5}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_8}(\lambda)$.

Game 9 In this game, we replace the key k_{A} derived in the test session π^s_i with the uniformly random and independent value $\widetilde{k_{\mathsf{A}}}$. We do so by defining a reduction $\mathcal{B}_6$ that interacts with an ind-cca KEM challenger $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, who replaces the tpk_{A} value sent in m_2 with the public key pk received from $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. When potential partner π^t_j should compute the ciphertext $ctxt_{\mathsf{A}}$ sent in m_{ack}, $\mathcal{B}_6$ instead replaces the computation of $ctxt_{\mathsf{A}}$ and k_{A} with the outputs of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, $\widetilde{ctxt_{\mathsf{A}}}$ and $\widetilde{k_{\mathsf{A}}}$ respectively. Whenever

party j requires the use of the secret key to decapsulate a ciphertext $ctxt'_{\mathsf{A}} \neq \widetilde{ctxt_{\mathsf{A}}}$, $\mathcal{B}_6$ simply queries $ctxt'_{\mathsf{A}}$ to $\mathcal{C}_{\mathsf{ind\text{-}cca}}$, and replaces the computation of k' with the output of $\mathcal{C}_{\mathsf{ind\text{-}cca}}$. By **Game 4** the public-key in m_2 was accepted by the source session without modification, and in **Game 11** we prove that the ciphertext in m_{ack} is accepted by π_i^s without modification. Detecting the replacement of k_{A} implies an efficient distinguishing QPT algorithm $\mathcal{A}$ against ind-cca security of KEM. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_8}(()\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_6}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda)$.

Game 10 Similar to **Game 3** in this game the challenger replaces the derived keys $mk_2, ck_2 = \mathsf{KDF}(\widetilde{k_{\mathsf{A}}}, \widetilde{ck_1})$ with uniformly random values $\widetilde{mk_2}, \widetilde{ck_2} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ by defining a reduction $\mathcal{B}_7$ that initializes a dual-prf challenger $\mathcal{C}_{\mathsf{dual\text{-}prf}}$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_7}(+)\mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda)$.

Game 11 Identical to **Game 4**, in this game we introduce an abort event that triggers if $\mathcal{A}$ is able to successfully forge τ_2. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_8}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_{11}}(\lambda)$. Note that by **Game 11** π_i^s aborts before accepting without a UT-match and thus, $\mathsf{Adv}^{\mathcal{A}}_{G_{11}}(\lambda) = 0$.

We now transition to **Case 2**.

Case 2: Test session does not UT-match another session, Compromise(i,s) nor Compromise(l,r) (such that $\pi_i^s.ck = \pi_l^r.ck$) have been issued before $\pi_i^s.\alpha =$ accept, Corrupt(i) has not been issued for S's long-term key before $\pi_i^s.\alpha =$ accept and $\pi_i^s.\rho =$ T..

We note that the proof of **Case 2** proceeds similarly to **Case 1**, but does not require steps to ensure that the ciphertext $ctxt_{\mathsf{A}}$ sent in $m_{\mathtt{fin}}$ reaches its U peer. Thus, we give a brief description of the change in each game, and point the reader to **Case 1** for details.

Game 0 This is the KIND security game and $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A},\mathbf{C2}}_{\mathsf{PQAG\text{-}HO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 In this game, we guess the index of the test session (i,s) and the intended source partner j and target partner l and abort if, during the execution of the experiment, a query $\mathsf{Test}(i',s')$ is received to a session $\pi_{i'}^{s'}$ such that $\pi_{i'}^{s'}.tpid = j', \pi_{i'}^{s'}.spid = l'$ and $(i,s,j,l) \neq (i',s',j',l')$. Thus $< \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq {n_P}^3 \cdot n_S \cdot < \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda)$.

Game 2 Identical to **Case 1 Game 2** in this game we replace ck computed in the test session π_i^s and its honest contributive keyshare session partner with a random key $\widetilde{ck}$ from the same key distribution by interacting with a hake challenger. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathsf{hake}}_{\mathsf{HAKE},\mathcal{B}_9}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda)$.

Game 3 In this game we replace the computation of the extracted keys $mk_0, ck_0 \xleftarrow{\$} \mathsf{KDF}(\widetilde{ck}, \epsilon)$ with a uniformly random and independent value $\widetilde{mk_0}, \widetilde{ck_0} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{10}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda)$.

Game 4 In this game we introduce an abort event if $\mathcal{A}$ can successfully forge τ_0 to the test session π^s_i by interacting with a sufcma MAC challenger. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_{11}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda)$.

Game 5 Identical to **Case 1 Game 2** in this game we replace gk_0 computed in the test session π^s_i and its honest contributive keyshare session partner with a random key $\widetilde{gk_0}$ from the same key distribution by interacting with a hake challenger. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda) \leq \mathsf{Adv}^{\mathsf{hake}}_{\mathsf{HAKE},\mathcal{B}_{12}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda)$.

Game 6 Here we abort if a hash collision occurs, by computing all hash values honestly and aborting if there exists two evaluations $(m, \mathsf{Hash}(m)), (m', \mathsf{Hash}(m'))$ such that $m \neq m'$ but $\mathsf{Hash}(m) = \mathsf{Hash}(m')$. The challenger $\mathcal{C}$ interacts with a Hash-collision challenger $\mathcal{C}_{\mathsf{coll}}$, outputting the collision if found. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) \leq \mathsf{Adv}^{\mathsf{coll}}_{\mathsf{Hash},\mathcal{C}_{\mathsf{coll}}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda)$.

Game 7 Identical to **Case 1 Game 7** in this game we abort if $\mathcal{A}$ is able to produce a value $c \leftarrow \mathsf{AEAD.Enc}(\widetilde{gk_0}, (ck_0 \| tpk_{\mathsf{A}} \| pk_{\mathsf{A}}), r_{\mathsf{G}_1}, \mathsf{T}_0 \| \mathsf{T}'_0)$ that decrypts correctly using $\widetilde{gk_0}$ by interacting with an AEAD-auth challenger. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) \leq \mathsf{Adv}^{\mathsf{auth}}_{\mathsf{AEAD},\mathcal{B}_{13}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda)$.

Game 8 Identical to **Case 1 Game 9** in this game, we replace the key k_{G_2} derived in the test session π^s_i with the uniformly random and independent value $\widetilde{k_{\mathsf{G}_2}}$ by interacting with an ind-cca KEM challenger. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_{14}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_8}(\lambda)$.

Game 9 In this game we replace the computation of the extracted keys $mk_1, ck_1 \xleftarrow{\$} \mathsf{KDF}(\widetilde{k_{\mathsf{G}_2}}, \widetilde{ck_0})$ with a uniformly random and independent value $\widetilde{mk_1}, \widetilde{ck_1} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_8}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{15}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda)$.

Game 10 In this game we abort if $\mathcal{A}$ can successfully forge τ_1 to the test session π^s_i by interacting with a sufcma MAC challenger. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_{16}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda)$. Since by **Game 10** π^s_i aborts when verifying a forged τ_1, $\mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda) = 0$.

We now transition to **Case 3**.

Case 3: Test session UT-match another session, $\mathsf{Compromise}(i,s)$ and $\mathsf{Compromise}(l,r)$ (such that $\pi^s_i.ck = \pi^r_l.ck$) were never issued. Since the majority

of game hops are similar to those in **Case 1**, we omit full details of each reduction and point readers to **Case 1**.

Game 0 This is the KIND security game and $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A},\mathbf{C3}}_{\mathsf{PQAG\text{-}HO},n_P,n_S}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 In this game, we guess the index of the test session (i, s) and the intended source partner j and target partner l and abort if, during the execution of the experiment, a query $\mathsf{Test}(i', s')$ is received to a session $\pi^{s'}_{i'}$ such that $\pi^{s'}_{i'}.tpid = j', \pi^{s'}_{i'}.spid = l'$ and $(i, s, j, l) \neq (i', s', j', l')$. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq {n_P}^3 \cdot n_S \cdot \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda)$.

Game 2 Identical to **Case 1 Game 9** in this game, we replace the key k_{A} derived in the test session π^s_i with the uniformly random and independent value $\widetilde{k_{\mathsf{A}}}$ by interacting with an ind-cca KEM challenger. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathsf{KEM},\mathcal{B}_{17}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda)$.

Game 3 In this game we replace the computation of the extracted keys $mk_2, ck_2 \xleftarrow{\$} \mathsf{KDF}(\widetilde{k_{\mathsf{A}}}, ck_1)$ with a uniformly random and independent value $\widetilde{mk_2}, \widetilde{ck_2} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{18}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda)$.

Game 4 In this game we abort if $\mathcal{A}$ can successfully forge τ_2 to the test session π^s_i by interacting with a sufcma MAC challenger. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) \leq \mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{B}_{19}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda)$.

Game 5 In this game we replace the computation of the session key $k \xleftarrow{\$} \mathsf{KDF}(ck_1, \widetilde{ck_2})$ with a uniformly random and independent value $\widetilde{k} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_4}(\lambda) \leq \mathsf{Adv}^{\mathsf{dual\text{-}prf}}_{\mathsf{PRF},\mathcal{B}_{20}}(\lambda) + \mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda)$. We emphasize that as a result of these changes, the session key $\widetilde{k}$ is now both uniformly random and independent of the protocol execution regardless of the bit b sampled by $\mathcal{C}$, thus $\mathcal{A}$ has no advantage in guessing the bit b. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) = 0$.

□

7.6 Conclusion

In this chapter, we presented PQAG-HO, a post-quantum secure handover protocol tailored for aviation. Our construction ensures the secure transition of an active CPDLC session as an aircraft moves between ground stations, addressing the challenges of resource-constrained avionic environments. We verify the security guarantees of PQAG-HO through formal security proofs, including notions of perfect forward secrecy and post compromise security, against quantum adversaries.

By integrating our HO handover and KEX key exchange formalisms, PQAG-HO demonstrates the practical application of these frameworks to create robust and domain-specific

protocols. We further instantiated the protocol using the post-quantum Kyber algorithm, highlighting its applicability within real-world infrastructures. We believe this work highlights the feasibility of bridging theoretical cryptographic formalisms with practical requirements for secure and reliable avionic communication in a post-quantum era.

Part V

Conclusions

Chapter 8

Conclusion

In this chapter, we summarize the key contributions of this thesis and outline potential directions for future research. We conclude this chapter with reflections on how our research as a whole advances security in the aerospace domain.

8.1 Conclusions & Future Work

The digitization of aerospace communication has rapidly evolved in the last decade. However, the security of communication protocols deployed within the aerospace domain remains rather immature and under-analyzed to withstand the threats posed by the current advanced cyber threat landscape. In this thesis, we set out to address this gap in three intertwined parallels.

Formalism for Handovers

In Part II we introduced the first formalism that recognizes handovers as a unique and standalone primitive. Handovers play an essential role within modern communication schemes; especially within infrastructure such as aviation that involve mobile entities that require maintaining a secure session as they geographically traverse between locations. Moreover, our formalism provides a generic and modular structure that can be leveraged to design and model secure handover schemes for any given domain. We further identified and formalized security properties that are desired or can be captured within a handover scheme and introduced StrongHO capturing all our identified notions of security. However, despite its strengths, our HO framework is not without limitations.

Particularly, the current HO framework does not explicitly capture session continuity. As a notion, session continuity enables concretely capturing the continuity of an existing session, as user U transitions between source S and target T, by maintaining verifiable session transcripts. This could provide additional handover-specific integrity guarantees for secure HO constructions. Moreover, our HO formalism only implicitly models *trusted* nodes, such as those represented by 5G home networks, which may limit its applicability in some contexts. Lastly, our HO model assumes honest S and T nodes for all captured security notions except for *path privacy*. Naturally, this assumption may not hold in real-world adversarial settings where compromised nodes pose significant risks. Extending our HO construction to capture key indistinguishability against a corrupt S node maybe desirable within this context. Strengthening our framework in this manner will expand its ability to capture security for more real-world deployments, such as when a 5G user U needs to transition from a corrupt home network S to an honest home network T.

Nevertheless, we believe the modularity and flexibility of our framework open many new avenues for research. Our formalization can be leveraged to analyze the security of known HO constructions such as 5G-HO and the proposed secure LDACS-HO for aviation. Furthermore, we believe our framework can easily be extended to capture the security of other known constructions such as OAuth and eduroam. To elaborate, we argue that OAuth can be considered a secure handover construction because it facilitates the delegation of access between a client, resource owner, and resource server through an intermediary authorization server. The authorization server negotiates secure access by issuing tokens that authenticate and authorize the client to access resources, similar to how the core network (S) in 5G securely negotiates a session handover from a user device (U) to a new base station (T). This intermediary-driven process ensures the continuity and security of the session during the transition of control between parties, making it well-suited for modeling and analysis within our HO formalism.

Analysis of Non-terrestrial Protocols

Space networking follows different restrictions than standard, terrestrial Internet-style networking, which as consequently motivated work in developing and deploying suitable networking protocols over the years. However, the cryptographic security of such, including DTN protocols like BPSec, has largely escaped formal cryptographic analysis, leading to a lack of understanding of how well the security intentions of such protocols are realized.

In Part III of this thesis, we provided the first formal cryptographic analysis and provable security treatment of one such protocol and lays a ground work for further analyses in this area. Furthermore, we propose StrongBPSec, a strong alternative to BPSec with additional integrity guarantees. We note that integrating our StrongBPSec with read receipts into existing BPSec instantiations and implementations can be achieved through multiple approaches: the existing standard could be extended with a new read receipt block type (e.g. as an Other Security Block [28, Section 10]) ; or, more conveniently, read receipts may also be introduced as an extension to the existing BIB block type in the form of a specialized BIB block. We leave exploring the specifics of how StrongBPSec can be incorporated within the existing BPSec paradigm to future work. We highlight, regardless of how it is integrated, the true value of StrongBPSec lies in adopting its read receipts with additional integrity guarantees within the current BPSec and BPSec Default Security Context standards. We further leave finding more efficient *key-wrapping* solutions and reducing the security overhead of BPSec to future work. Additionally, we identify significant opportunities to enhance the current BPSec instantiation by incorporating stronger notions of security, such as forward secrecy and post-compromise security, within its construction.

Secure Protocols for Avionic Communication

Part IV Chapter 5 of this thesis briefly formalizes key exchange schemes as KEX. Then we integrate our KEX formalism within the HAKE model and provided detailed algorithmic description about the modified and simplified HAKE framework. We leverage this modified HAKE model for the analysis of all our protocols proposed in Chapters 6 and 7.

In Part IV Chapter 6, we proposed a pair of quantum-secure hybrid key exchange protocols PQAG-KEM and PQAG-SIG for securing communication in avionic infrastructure. We formally verify the security of both protocols in our modified HAKE model, against future-quantum as well as classical attackers, highlighting its suitability to provide long-term security within critical national infrastructures. We benchmarked our protocols with different post-quantum algorithms and compared their performance against other state-of-the-art avionic communication protocols and illustrated that our protocols combines tight security with fast performance even within resource-constrained environments.

PQAG opens up many avenues for future work. First, instantiating PQAG with other post-quantum and classical cryptographic schemes to explore even more efficient combinations suitable for resource-constrained real-world application. For instance, combining

Kyber with gap-Diffie-Hellman is likely to produce significantly faster performance even with real-time key generation. This combination will also aid in reducing the communication cost by further shortening the size of the classical key exchange components. Moreover, comparing PQAG-KEM and PQAG-SIG with STS-CSIDH by simulating all protocols within a realistic network environment, capturing the constraints of data links deployed within aviation infrastructures would further demonstrate the practicality of post-quantum cryptography within aviation infrastructure. In addition, PQAG needs to be extended so that it can also facilitate the verification of transcript consistency between aircrafts (A) and ground stations (G) as the flight moves from one G to another. The primary purpose of this extension would be to assist the handover of communication between As and Gs as an aircraft moves from one geographical location to another. We explore this last direction in the final chapter of our thesis.

Lastly, in Part IV Chapter 7, we introduced PQAG-HO, a quantum-secure avionic handover scheme designed leveraging our formalism proposed in II. We designed PQAG-HO to capture notions of security desirable within a commercial avionic setting, including verifiable session transcripts to guarantee the integrity of the session after a handover has completed. Moreover, we instantiated our construction using a combination of post-quantum and classical primitives and benchmarked its performance for walltime. At the time of writing, there were no other comparable implementations of such avionic handover schemes. While the work of [87] and [72] propose handover schemes for aviation, we highlight that they do not implement their proposed constructions and only the work of [87] provide formal proofs for forward secrecy and mutual authentication properties of their protocol. In contrast, we instantiate and implement our construction and provide comprehensive formal proofs for key-indistinguishability within our KEX and modified HAKE frameworks. We leave simulating our PQAG-HO construction within a realistic avionic environment and benchmarking its performance within the bandwidth constraints of avionic datalinks to future work.

8.2 Reflections

Overall, in the course of this thesis, we have explored multiple avenues that collectively endeavor to strengthen the security of existing communication protocols in the aerospace domain. We started our research by proposing a generic handover construction that can be leveraged to both analyze the security of existing avionic handovers such as LDACS-HO [87] or design new schemes as our proposed PQAG-HO 7. However, the modularity

and flexibility of our handover construction extend beyond aviation, and enables its usage within other contexts where strong security of handovers is desired.

Moreover, in Chapter 3, we argue that secure handovers (SHO) should be recognized as a distinct cryptographic primitive, separate from traditional key exchange constructions (KEX). A central justification for this distinction is our introduction of *path privacy*—a security property that can only be meaningfully captured when handovers are modeled as a standalone primitive. However, we acknowledge that it is neither appropriate nor necessary for all SHO schemes to capture *path privacy*, and the modularity of our framework explicitly allows for this flexibility.

For instance, while concealing its flight path may be appropriate for military aircraft facing adversarial insider threats, civil aviation standards prioritize safety and reliability, mandating continuous tracking of commercial aircraft. This practical constraint is reflected in our own work: the StrongHO construction in Figure 3.7 exemplifies a generic SHO construction, achieving the highest level of security within our model—including *path privacy*—whereas our post-quantum secure avionic SHO scheme, PQAG-HO, introduced in Chapter 7, intentionally omits *path privacy* due to aviation-specific requirements.

We also acknowledge the additional complexity introduced by incorporating post-quantum primitives as foundational components within SHO constructions. For example, our proposed PQAG-HO, while not capturing *path privacy*, still incurs significant processing, memory, and bandwidth overheads—costs that would likely increase further were *path privacy* to be supported.

Counterintuitively, these observations may appear to weaken our argument for treating SHO as a distinct primitive, an argument that was strongly grounded in the premise that *path privacy* is a critical and uniquely SHO-specific property. Our response to this apparent paradox is twofold. First, in decentralized and heterogeneous networks (e.g., 5G, IoT), users rely on infrastructure operated under varying trust assumptions. In such settings, adversarial insiders pose a significant threat. Modeling handovers as a dedicated primitive enables us to formalize and reason about such privacy threats during the handover phase—threats that are not adequately addressed by traditional KEX models that do not recognize a mobile user wishing to securely transition its existing session from one party to another. Second, while *path privacy* introduces complexity—particularly in post-quantum constructions—it remains a critical property in high-risk domains and should be supported where applicable. Designing SHO schemes that support *path privacy* in constrained real-world environments remains an open challenge, which we leave to future work.

We now address the second point: whether a SHO that omits *path privacy* can be effectively modeled as a proxied KEX. While such abstraction may constrain extensibility—particularly the ability to retrofit *path privacy*—it remains a viable modeling approach. Crucially, the security properties of *key indistinguishability* and *unlinkability*, formalized in Chapter 3, are commonly captured within existing KEX frameworks. In fact, the security of our SHO *unlinkability* notion is upper-bounded by the indistinguishability of the underlying keys that link a user from one session to the next. Accordingly, proxied KEX constructions may suffice in domains where *path privacy* is not a critical concern, offering an efficient and semantically aligned alternative—while still fitting within the broader design space articulated by our framework.

Therefore, depending on domain-specific constraints, required security guarantees, and the need for future adaptability, SHOs may be modeled either as a distinct primitive—offering fine-grained control and stronger security—or as a proxied KEX construction that favors simplicity and conventional security guarantees.

Next, we formally analyzed the security of BPSec, a deep space communication protocol tailored to withstand the abnormalities of the non-terrestrial Internet. Our analysis of BPSec and the proposed security enhancements is the first of its kind, highlighting critical security implications in a domain that has been increasingly in the spotlight, particularly with the growing interest in commercial space programs like SpaceX. We then turned our attention to secure avionic communication, first proposing a pair of key-exchange protocols for aviation, PQAG-SIG and PQAG-KEM. Designing and instatiating two protocols built on distinct separate cryptographic primitives allowed us to evaluate their suitability for the domain-specific constraints of aviation. Moreover, we designed our constructions with future-proof security in mind, combining classical and post-quantum primitives to provide hybrid notions of security. Our approach addressed both the threat of quantum adversaries to classical cryptosystems and the potential vulnerabilities of post-quantum primitives. We provided formal proof for both our constructions, and analyzed their security in our modified HAKE model, against both classical and quantum attackers. Lastly, we proposed a post-quantum secure handover scheme PQAG-HO for aviation, designed using our generic handover framework HO. We highlight that PQAG-HO is the first such instantiation of a handover scheme designed within our HO framework introduced in II, and its successful implementation and benchmarking reaffirm the practical applicability of our universal HO model.

Bibliography

[1] CCSDS File Delivery Protocol (CFDP). Technical Report CSDS 727.0-B-5, The Consultative Committee for Space Data Systems. Recommended Standard., 2020. URL `https://public.ccsds.org/Pubs/727x0b5.pdf`.

[2] Space Packet Protocol. Technical Report CSDS 133.0-B-2, The Consultative Committee for Space Data Systems. Recommended Standard, 2020. URL `https://public.ccsds.org/Pubs/133x0b2e1.pdf`.

[3] Overview of Space Communications Protocols. Technical Report CCSDS 130.0-G-4, Consultative Committee for Space Data Systems, Washington DC, USA, Apr. 2023. URL `https://public.ccsds.org/Pubs/130x0g4e1.pdf`.

[4] E. . Cybersecurity in aviation | eurocontrol, 08 2019. URL `https://www.eurocontrol.int/publication/cybersecurity-aviation`.

[5] N. Ahmad, H. Cruickshank, Z. Sun, and M. Asif. Pseudonymised communication in delay tolerant networks. In *2011 Ninth Annual International Conference on Privacy, Security and Trust*, pages 1–6, 2011. doi: 10.1109/PST.2011.5971956.

[6] R. Alnashwan, P. Gope, and B. Dowling. Privacy-aware secure region-based handover for small cell networks in 5g-enabled mobile communication. *IEEE Transactions on Information Forensics and Security*, 18:1898–1913, 2023.

[7] J. Alperin-sheriff, J. Kelsey, C. Miller, R. Peralta, and D. Smith-tone. Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. Technical report, 2020.

[8] G. Arfaoui, X. Bultel, P.-A. Fouque, A. Nedelcu, and C. Onete. The privacy of the tls 1.3 protocol. *Cryptology ePrint Archive*, 2019.

[9] A. Asari, M. R. Alagheband, M. Bayat, and M. R. Asaar. A new provable hierarchical anonymous certificateless authentication protocol with aggregate verification in ADS-B systems. *Computer Networks*, 2020. ISSN 13891286. doi: 10.1016/j.comnet.2020.107599.

[10] N. Asokan, K. Kostiainen, P. Ginzboorg, J. Ott, and C. Luo. Applicability of identity-based cryptography for disruption-tolerant networking. In *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking*, MobiOpp '07, page 52–56. Association for Computing Machinery, 2007. ISBN 9781595936882. doi: 10.1145/1247694.1247705. URL `https://doi-org.libproxy.nps.edu/10.1145/1247694.1247705`.

[11] P. C. Authority. Authenticated encryption — cryptography 2.7.dev1 documentation, 2013. URL `https://cryptography.io/en/latest/hazmat/primitives/aead/`.

[12] N. Aviram, K. Gellert, and T. Jager. Session resumption protocols and efficient forward security for tls 1.3 0-rtt. *Journal of Cryptology*, 34(3):20, 2021.

[13] M. Backendal, F. Günther, and K. G. Paterson. Puncturable key wrapping and its applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 651–681. Springer, 2022.

[14] C. Badertscher, C. Matt, U. Maurer, P. Rogaway, and B. Tackmann. Augmented Secure Channels and the Goal of the TLS 1.3 Record Layer. In *Proceedings of the 9th International Conference on Provable Security - Volume 9451*, ProvSec 2015, page 85–104. Springer-Verlag, 2015. ISBN 9783319260587. doi: 10.1007/978-3-319-26059-4_5. URL `https://doi.org/10.1007/978-3-319-26059-4_5`.

[15] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler. A formal analysis of 5g authentication. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1383–1396, 2018.

[16] C. Battarbee, C. Striecks, L. Perret, S. Ramacher, and K. Verhaeghe. Quantum-safe hybrid key exchanges with kem-based authentication. *arXiv preprint arXiv:2411.04030*, 2024.

[17] M. Bellare and A. Lysyanskaya. Symmetric and Dual PRFs from Standard Assumptions: A Generic Validation of an HMAC Assumption. *IACR Cryptology ePrint Archive*, 2015:1198, 2015. URL `http://dblp.uni-trier.de/db/journals/iacr/iacr2015.html{#}BellareL15`.

[18] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 531–545. Springer, 2000.

[19] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Annual international cryptology conference*, pages 232–249. Springer, 1993.

[20] M. Bellare and P. Rogaway. Provably secure session key distribution: the three party case. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 57–66, 1995.

[21] M. Bellare, T. Kohno, and C. Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS)*, pages 1–11, 2002.

[22] M. A. Bellido-Manganell, T. Gräupl, O. Heirich, N. Mäurer, A. Filip-Dhaubhadel, D. M. Mielke, L. M. Schalk, D. Becker, N. Schneckenburger, and M. Schnell. Ldacs flight trials: Demonstration and performance analysis of the future aeronautical communications system. *IEEE Transactions on Aerospace and Electronic Systems*, 2021.

[23] F. Z. Benhamida, A. Bouabdellah, and Y. Challal. Using delay tolerant network for the Internet of Things: Opportunities and challenges. In *2017 8th International Conference on Information and Communication Systems (ICICS)*, pages 252–257, 2017. doi: 10.1109/IACS.2017.7921980.

[24] W. Beullens. Breaking rainbow takes a weekend on a laptop. *Cryptology ePrint Archive*.

[25] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. In *Post-Quantum Cryptography: 10th International Conference, PQCrypto 2019, Chongqing, China, May 8–10, 2019 Revised Selected Papers 10*, pages 206–226. Springer, 2019.

[26] N. Bindel, J. Brendel, M. Fischlin, and D. Stebila. *Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange*. Number PQCrypto. 2019. ISBN 9783030255107. doi: 10.1007/978-3-030-25510-7.

[27] E. Birrane. *Securing Delay-Tolerant Networks with BPSec*. Wiley, 2023. URL `https://ieeexplore.ieee.org/servlet/opac?bknumber=10015530`.

[28] E. J. Birrane and K. McKeever. Bundle Protocol Security (BPSec). RFC 9172, Jan. 2022. URL `https://www.rfc-editor.org/info/rfc9172`.

[29] E. J. Birrane, A. White, and S. Heiner. Default Security Contexts for Bundle Protocol Security (BPSec). RFC 9173, Jan. 2022. URL `https://www.rfc-editor.org/info/rfc9173`.

[30] O. Blazy, I. Boureanu, P. Lafourcade, C. Onete, and L. Robert. How fast do you heal? a taxonomy for post-compromise security in secure-channel establishment. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 5917–5934, 2023.

[31] C. Boyd and B. Hale. Secure Channels and Termination: The Last Word on TLS. In *Latincrypt*, 2017. URL `https://api.semanticscholar.org/CorpusID:3648242`.

[32] C. Boyd and W. Mao. On a limitation of ban logic. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 240–247. Springer, 1993.

[33] C. Boyd, B. Hale, S. F. Mjølsnes, and D. Stebila. From Stateless to Stateful: Generic Authentication and Authenticated Encryption Constructions with Application to TLS. In *Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610*, page 55–71, 2016. ISBN 9783319294841. doi: 10.1007/978-3-319-29485-8_4. URL `https://doi.org/10.1007/978-3-319-29485-8_4`.

[34] S. Bruckner, S. Ramacher, and C. Striecks. Muckle+: End-to-end hybrid authenticated key exchanges. In *International Conference on Post-Quantum Cryptography*, pages 601–633. Springer, 2023.

[35] S. Burleigh, K. Fall, and E. J. Birrane. Bundle Protocol Version 7. RFC 9171, Jan. 2022. URL `https://www.rfc-editor.org/info/rfc9171`.

[36] S. C. Burleigh, S. Farrell, and M. Ramadas. Licklider Transmission Protocol - Specification. RFC 5326, Sept. 2008. URL `https://www.rfc-editor.org/info/rfc5326`.

[37] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems (TOCS)*, 8(1):18–36, 1990.

[38] M. Campagna. Preparing today for a post-quantum cryptographic future, 07 2022. URL `https://www.amazon.science/blog/preparing-today-for-a-post-quantum-cryptographic-future`.

[39] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *International conference on the theory and applications of cryptographic techniques*, pages 453–474. Springer, 2001.

[40] J. Cao, M. Ma, Y. Fu, H. Li, and Y. Zhang. Cppha: Capability-based privacy-protection handover authentication mechanism for sdn-based 5g hetnets. *IEEE transactions on dependable and secure computing*, 18(3):1182–1195, 2019.

[41] W. Castryck and T. Decru. An efficient key recovery attack on sidh (preliminary version). *Cryptology ePrint Archive*, 2022.

[42] W. Castryck and T. Decru. An efficient key recovery attack on sidh (preliminary version). *Cryptology ePrint Archive*, 2022.

[43] J. Chu. The beginning of the end for encryption schemes?, 2016.

[44] CISA. Critical infrastructure sectors, 2022. URL `https://www.cisa.gov/topics/critical-infrastructure-security-and-resilience/critical-infrastructure-sectors`.

[45] S. Coretti, U. Maurer, and B. Tackmann. A constructive perspective on key encapsulation. *Number Theory and Cryptography: Papers in Honor of Johannes Buchmann on the Occasion of His 60th Birthday*, pages 226–239, 2013.

[46] J. Cox. "i gave a bounty hunter $300. then he located our phone". *Vice Media Group. URL: https://motherboard. vice. com/en_us/article/nepxbz/i-gave-a-bountyhunter-300-dollars-located-phone-microbilt-zumigo-tmobile*, 2019.

[47] C. Cremers and M. Feltz. Beyond eck: Perfect forward secrecy under actor compromise and ephemeral-key reveal. In *European Symposium on Research in Computer Security*, pages 734–751. Springer, 2012.

[48] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 1976. ISSN 15579654. doi: 10.1109/TIT.1976.1055638.

[49] W. Diffie, P. C. Van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and cryptography*, 2(2):107–125, 1992.

[50] B. Dowling, T. B. Hansen, and K. G. Paterson. Many a mickle makes a muckle: A framework for provably quantum-secure hybrid key exchange. In *International Conference on Post-Quantum Cryptography*, pages 483–502. Springer, 2020.

[51] M. Dworkin. Request for Review of Key Wrap Algorithms. Cryptology ePrint Archive, Paper 2004/340, 2004. URL `https://eprint.iacr.org/2004/340`. `https://eprint.iacr.org/2004/340`.

[52] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003.

[53] C.-I. Fan, J.-J. Huang, M.-Z. Zhong, R.-H. Hsu, W.-T. Chen, and J. Lee. Rehand: Secure region-based fast handover with user anonymity for small cell networks in mobile communications. *IEEE Transactions on Information Forensics and Security*, 15:927–942, 2019.

[54] M. Fischlin, F. Günther, G. A. Marson, and K. G. Paterson. Data is a stream: Security of stream-based channels. In *Advances in Cryptology–CRYPTO 2015, Proceedings, Part II 35*, pages 545–564. Springer, 2015.

[55] M. Fischlin, F. Günther, and C. Janson. Robust Channels: Handling Unreliable Networks in the Record Layers of QUIC and DTLS 1.3. *J. Cryptol.*, 37(2), jan 2024. ISSN 0933-2790. doi: 10.1007/s00145-023-09489-9. URL `https://doi.org/10.1007/s00145-023-09489-9`.

[56] S. Frankel and S. Krishnan. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap. RFC 6071, Feb. 2011. URL `https://datatracker.ietf.org/doc/html/rfc6071`.

[57] B. Fung. Fcc fines wireless carriers millions for sharing user locations without consent. *CNN*, Apr. 2024. URL `https://edition.cnn.com/2024/04/29/tech/fcc-fines-att-verizon-200-million/index.html`.

[58] I. Goldberg, D. Stebila, and B. Ustaoglu. Anonymity and one-way authentication in key exchange protocols. *Designs, Codes and Cryptography*, 67(2):245–269, 2013.

[59] F. Günther and S. Mazaheri. A Formal Treatment of Multi-key Channels. In *Advances in Cryptology–CRYPTO 2017 Proceedings, Part III 37*, pages 587–618. Springer, 2017.

[60] S. Gupta, B. L. Parne, and N. S. Chaudhari. Security vulnerabilities in handover authentication mechanism of 5g network. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 369–374. IEEE, 2018.

[61] S. Gupta, B. L. Parne, N. S. Chaudhari, and S. Saxena. Seai: Secrecy and efficiency aware inter-gnb handover authentication and key agreement protocol in 5g communication network. *Wireless Personal Communications*, 122(4):2925–2962, 2022.

[62] S. Herwig. Towards protecting billions and billions of bits on the interplanetary internet. In *SpaceSec*, 2023.

[63] R. Housley and J. Schaad. Advanced Encryption Standard (AES) Key Wrap Algorithm. RFC 3394, Oct. 2002. URL `https://www.rfc-editor.org/info/rfc3394`.

[64] J. Huang and Y. Qian. A secure and efficient handover authentication and key management protocol for 5g networks. *Journal of Communications and Information Networks*, 5(1):40–49, 2020.

[65] ICAO. Global operational data link document (gold) manual. pages 24–34. International Civil Aviation Organization, 2013. `https://www.icao.int/APAC/Documents/edocs/GOLD_2Edition.pdf`.

[66] A. Jones. Europe launches ambitious 'moonlight' program to support lunar exploration, 10 2024. URL `https://www.space.com/europe-moonlight-program-lunar-navigation-communications#`.

[67] P. Kampanakis, T. Hansen, A. Volanis, and G. Ravago. Post-quantum hybrid sftp file transfers using aws transfer family | amazon web services, 06 2023. URL `https://aws.amazon.com/blogs/security/post-quantum-hybrid-sftp-file-transfers-using-aws-transfer-family/`.

[68] A. Kate, G. M. Zaverucha, and U. Hengartner. Anonymity and security in delay tolerant networks. In *2007 Third International Conference on Security and Privacy*

in Communications Networks and the Workshops - SecureComm 2007, pages 504–513, 2007. doi: 10.1109/SECCOM.2007.4550373.

[69] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. CRC press, 2014. ISBN 1351133012.

[70] C. F. Kerry and P. D. Gallagher. Digital signature standard (dss). *FIPS PUB*, pages 186–4, 2013.

[71] S. Khan, A. Gurtov, A. Braeken, and P. Kumar. A Security Model for Controller-Pilot Data. In *Integrated Communications Navigation and Surveillance Conference (ICNS)*, pages 1–10, 2021. ISBN 9781665435840.

[72] S. Khan, G. S. Gaba, A. Gurtov, L. J. Jansen, N. Mäurer, and C. Schmitt. Post quantum secure handover mechanism for next generation aviation communication networks. *IEEE Transactions on Green Communications and Networking*, 2024.

[73] H. Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In *Annual Cryptology Conference*, pages 631–648. Springer, 2010.

[74] H. Krawczyk. Cryptographic extraction and key derivation: The hkdf scheme. In *Annual Cryptology Conference*, pages 631–648. Springer, 2010.

[75] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication, 1997.

[76] D. Kwon, S. Son, Y. Park, H. Kim, Y. Park, S. Lee, and Y. Jeon. Design of secure handover authentication scheme for urban air mobility environments. *IEEE Access*, 10:42529–42541, 2022.

[77] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *International conference on provable security*, pages 1–16. Springer, 2007.

[78] B. LaMacchia, K. Lauter, and A. Mityagin. Stronger security of authenticated key exchange. In *International conference on provable security*, pages 1–16. Springer, 2007.

[79] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 147–166. Springer, 1996.

[80] X. Lv, Y. Mu, and H. Li. Non-Interactive Key Establishment for Bundle Security Protocol of Space DTNs. *IEEE Transactions on Information Forensics and Security*, 9(1):5–13, 2014. doi: 10.1109/TIFS.2013.2289993.

[81] V. Lyubashevsky, S. Bai, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, and D. Stehlé. Crystals-dilithium. 2022. `https://openquantumsafe.org/liboqs/algorithms/sig/dilithium`.

[82] L. Maino and C. Martindale. An attack on sidh with arbitrary starting curve. *Cryptology ePrint Archive*, 2022.

[83] C. G. Manning. Delay/Disruption Tolerant Networking Overview, 2023. URL `https://www.nasa.gov/technology/space-comms/delay-disruption-tolerant-networking-overview/`.

[84] C. G. Manning. Frequently Asked Questions, 2023. URL `https://www.nasa.gov/technology/space-comms/delay-disruption-tolerant-networking-faq/`.

[85] T. Maremont, Mark McGinty. Ready for Departure: M&A Airlines. *The Wall Street Journal*, 2011. URL `https://www.wsj.com/articles/SB10001424052702303499204576389923856575528`.

[86] N. Mäurer, T. Gräupl, C. Gentsch, and C. Schmitt. Comparing different diffie-hellman key exchange flavors for ldacs. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–10. IEEE, 2020.

[87] N. Maurer, T. Graupl, C. Schmitt, C. Rihacek, and B. Haindl. A secure ground handover protocol for ldacs. In *Proceedings of International Workshop on ATM/CNS 2022 International Workshop on ATM/CNS*, pages 1–8. Electronic Navigation Research Institute, 2022.

[88] N. Mäurer, T. Gräupl, C. Schmitt, G. D. Rodosek, and H. Reiser. Advancing the security of ldacs. *IEEE Transactions on Network and Service Management*, 19(4): 5237–5251, 2022.

[89] S. A. Menesidou, V. Katos, and G. Kambourakis. Cryptographic Key Management in Delay Tolerant Networks: A Survey. *Future Internet*, 9(3):26, 2017.

[90] D. M. Mielke, N. Mäurer, T. Gräupl, and M. A. Bellido-Manganell. 1. quantum applications-fachbeitrag: Getting civil aviation ready for the post quantum age with ldacs. *Digitale Welt*, 5(4):28, 2021.

[91] R. Miller, I. Boureanu, S. Wesemeyer, and C. J. Newton. The 5g key-establishment stack: In-depth formal verification and experimentation. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, pages 237–251, 2022.

[92] A. Moore. I didn't want it anywhere near me': how the apple airtag became a gift to stalkers. *The Guardian*, 2022. `https://www.theguardian.com/technology/2022/sep/05/i-didnt-want-it-anywhere-near-me-how-theapple-airtag-became-a-gift-to-stalkers`.

[93] D. M. Nessett. A critique of the burrows, abadi and needham logic. *ACM SIGOPS Operating Systems Review*, 24(2):35–38, 1990.

[94] K. Norrman. Secure anycast channels with applications to 4g and 5g handovers. *Cryptology ePrint Archive*, 2022.

[95] V. O. Nyangaresi, A. J. Rodrigues, and S. O. Abeka. Machine learning protocol for secure 5g handovers. *International Journal of Wireless Information Networks*, 29(1): 14–35, 2022.

[96] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. Technical report, 1993.

[97] A. Peltonen, R. Sasse, and D. Basin. A comprehensive formal analysis of 5g handover. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 1–12, 2021.

[98] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *CryptoBytes*, 5(2):2–13, 2002.

[99] A. Pfitzmann and M. Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management, 2010.

[100] Phil Demetriou. pqcrypto 0.1.3, 2020. `https://pypi.org/project/pqcrypto/`.

[101] Python. hashlib — secure hashes and message digests — python 3.8.4rc1 documentation. URL `https://docs.python.org/3/library/hashlib.html`.

[102] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Technical report, Internet Engineering Task Force (IETF). RFC 8446, 2018.

[103] D. Robert. Breaking sidh in polynomial time. *Cryptology ePrint Archive*, 2022.

[104] F. Rodríguez-Henríquez. Sibc: A python-3 library for designing and implementing efficient isogeny-based protocols. 2021.

[105] P. Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 98–107. Association for Computing Machinery, 2002. ISBN 1581136129. doi: 10.1145/586110.586125. URL `https://doi.org/10.1145/586110.586125`.

[106] P. Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 98–107, 2002.

[107] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *Fast Software Encryption: 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004. Revised Papers 11*, pages 371–388. Springer, 2004.

[108] P. Rogaway and T. Shrimpton. A Provable-Security Treatment of the Key-Wrap Problem. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 373–390. Springer, 2006.

[109] S. Rüsch, D. Schürmann, R. Kapitza, and L. Wolf. Forward Secure Delay-Tolerant Networking. In *Proceedings of the 12th Workshop on Challenged Networks*, CHANTS '17, page 7–12. Association for Computing Machinery, 2017. ISBN 9781450351447. doi: 10.1145/3124087.3124094. URL `https://doi.org/10.1145/3124087.3124094`.

[110] A. Sahai and B. Waters. How to use indistinguishability obfuscation: Deniable encryption. Technical report, and more. Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint . . . , 2013.

[111] P. Schwabe, D. Stebila, and T. Wiggers. Post-quantum tls without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1461–1480, 2020.

[112] P. Schwabe, D. Stebila, and T. Wiggers. More efficient post-quantum kemtls with predistributed public keys. In *European Symposium on Research in Computer Security*, pages 3–22. Springer, 2021.

[113] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994. ISBN VO -. doi: 10.1109/SFCS.1994.365700.

[114] B. Sipos. DTN Bundle Protocol Security (BPSec) COSE Context. Internet-Draft draft-ietf-dtn-bpsec-cose-04, Internet Engineering Task Force, July 2024. URL `https://datatracker.ietf.org/doc/draft-ietf-dtn-bpsec-cose/04/`. Work in Progress.

[115] J. Smailes, D. Moser, M. Smith, M. Strohmeier, V. Lenders, and I. Martinovic. You talkin'to me? exploring practical attacks on controller pilot data link communications. In *Proceedings of the 7th ACM on Cyber-Physical System Security Workshop*, pages 53–64, 2021.

[116] M. Smith, M. Strohmeier, V. Lenders, and I. Martinovic. On the security and privacy of ACARS. In *ICNS 2016: Securing an Integrated CNS System to Meet Future Challenges*, 2016. ISBN 9781509021499. doi: 10.1109/ICNSURV.2016.7486395.

[117] S. Son, J. Lee, Y. Park, Y. Park, and A. K. Das. Design of blockchain-based lightweight v2i handover authentication protocol for vanet. *IEEE Transactions on Network Science and Engineering*, 9(3):1346–1358, 2022. doi: 10.1109/TNSE.2022.3142287.

[118] E. Stallard. Elon musk's starship rocket achieves record-breaking feat. *BBC*, 10 2024. URL `https://www.bbc.com/news/articles/c8xe7exjy1go`.

[119] M. Strohmeier, M. Schafer, M. Smith, V. Lenders, and I. Martinovic. Assessing the impact of aviation security on cyber power. In *International Conference on Cyber Conflict, CYCON*, 2016. ISBN 9789949954483. doi: 10.1109/CYCON.2016.7529437.

[120] P. Syverson, R. Dingledine, and N. Mathewson. Tor: The Second Generation Onion Router. In *Usenix Security*, pages 303–320. USENIX Association Berkeley, CA, 2004.

[121] The PyNaCl developers. PyNaCl 1.5.0, 2022. `https://pypi.org/project/PyNaCl/`.

[122] The Python Cryptographic Authority. cryptography 37.0.2. 2022. `https://pypi.org/project/cryptography/`.

[123] The White House. National security memorandum on promoting united states leadership in quantum computing while mitigating risks to vulnerable cryptographic systems. 2022. `https://www.whitehouse.gov/briefing-room/statements-releases/2022/05/04/national-security-memorandum-on-promoting-united-states-\leadership-in-quantum-computing-while-mitigating-risks-to-\vulnerable-cryptographic-systems/`.

[124] G. Thumbur, N. B. Gayathri, P. Vasudeva Reddy, M. D. Zia Ur Rahman, and A. Lay-Ekuakille. Efficient pairing-free identity-based ADS-B authentication scheme with batch verification. *IEEE Transactions on Aerospace and Electronic Systems*, 2019. ISSN 15579603. doi: 10.1109/TAES.2018.2890354.

[125] M. Tiepelt, C. Martin, and N. Maeurer. Post-quantum ready key agreement for aviation. *IACR Communications in Cryptology*, 1(1), 2024. ISSN 3006-5496. doi: 10.62056/aebn2isfg.

[126] L. Torgerson, S. C. Burleigh, H. Weiss, A. J. Hooke, K. Fall, D. V. G. Cerf, K. Scott, and R. C. Durst. Delay-Tolerant Networking Architecture. RFC 4838, Apr. 2007. URL `https://www.rfc-editor.org/info/rfc4838`.

[127] M. Wang, D. Zhao, Z. Yan, H. Wang, and T. Li. Xauth: Secure and privacy-preserving cross-domain handover authentication for 5g hetnets. *IEEE Internet of Things Journal*, 10(7):5962–5976, 2023. doi: 10.1109/JIOT.2022.3223223.

[128] K. D. Wesson, T. E. Humphreys, and B. L. Evans. Can Cryptography Secure Next Generation Air Traffic Surveillance. *IEEE Security & Privacy*, 2014.

[129] B. Westerbaan and C. D. Rubin. Defending against future threats: Cloudflare goes post-quantum. *The Cloudflare Blog*, 2022. URL `https://blog.cloudflare.com/post-quantum-for-all/`.

[130] H. Yang, Q. Zhou, M. Yao, R. Lu, H. Li, and X. Zhang. A practical and compatible cryptographic solution to ADS-B security. *IEEE Internet of Things Journal*, 2019. ISSN 23274662. doi: 10.1109/JIOT.2018.2882633.

[131] X. Yang, X. Huang, and J. K. Liu. Efficient handover authentication with user anonymity and untraceability for mobile cloud computing. *Future Generation Computer Systems*, 62:190–195, 2016.

[132] Z. Yang, Y. Chen, and S. Luo. Two-message key exchange with strong security from ideal lattices. In *Cryptographers' Track at the RSA Conference*, pages 98–115. Springer, 2018.

[133] Y. Zhang, R. H. Deng, E. Bertino, and D. Zheng. Robust and universal seamless handover authentication in 5g hetnets. *IEEE Transactions on Dependable and Secure Computing*, 18(2):858–874, 2019.

[134] J. Zhou, M. Song, J. Song, X.-W. Zhou, and L. Sun. Autonomic Group Key Management in Deep Space DTN. *Wirel. Pers. Commun.*, 77(1):269–287, July 2014. ISSN 0929-6212. doi: 10.1007/s11277-013-1505-1. URL `https://doi.org/10.1007/s11277-013-1505-1`.