Path Privacy and Handovers: Preventing Insider Traceability Attacks During Secure Handovers

Rabiah Alnashwan
School of Computer Science
University of Sheffield
Sheffield, United Kingdom
ralnashwan1@sheffield.ac.uk

Benjamin Dowling
Department of Informatics
King's College London
London, United Kingdom
benjamin.dowling@kcl.ac.uk

Bhagya Wimalasiri
School of Computer Science
University of Sheffield
Sheffield, United Kingdom
b.m.wimalasiri@sheffield.ac.uk

***Abstract*—The rise of 5G and IoT has shifted secure communication from centralized and homogeneous to a landscape of heterogeneous mobile devices constantly travelling between myriad networks. 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 formalisation of secure handover schemes. We leverage our formalisation 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.**

***Index Terms*—Secure Handover, Path Privacy, Formalisation, Provable Security, Protocol Analysis.**

I. Introduction

Secure handover protocols enable a secure communication session, previously established with a given communication partner, to securely transition to a second communication partner without loss of functionality or security. 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 [1] 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 a user's device moving between different zones [1].

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, eliminating the requirement to re-establish a new session every time an aircraft enters a subsequent geographic zone [2]. Figure 1 illustrates an expected implementation of a CPDLC connection handover as described by the official ICAO guidelines [2].

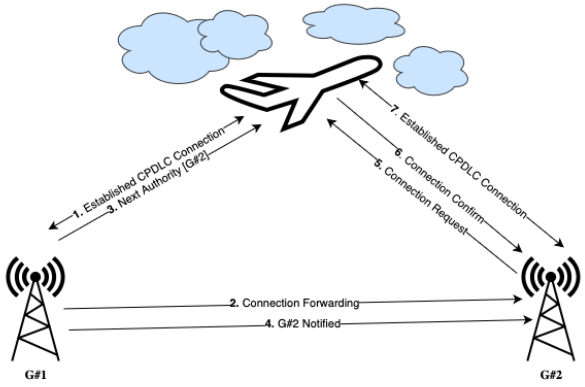

Fig. 1: A generic CPDLC handover between source ground station G#1, target ground station G#2 and an aircraft.

Despite numerous proposals for innovative handover schemes [3], there exists no formal framework defining handovers as a distinct primitive with unique functional requirements and security properties. Most literature on handover schemes is modelled after the 5G protocol [1], [4], with little discussion on systematically defining the functional goals of a handover or distinguishing handover schemes from other primitives like key exchange protocols. Similarly, little work has been done to formalise the security goals that handover protocols should achieve. Failing to understand and formalise the security of handover schemes leaves these protocols open to as-of-yet-undiscovered attacks.

Both 5G handover and CPDLC have suffered from various proposed attacks. Gupta et al. [5] highlight a desynchronisation attack on the 5G handover protocol using a rogue base station to enable denial-of-service attacks. Basin et al. [6] 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. [1] 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 lacks security guarantees as it operates over unencrypted and unauthenticated channels. Smailes et al. [7] demonstrated practical attacks on CPDLC, successfully impersonating ATC stations to aircrafts and hijacking a session during handover. They triggered false handovers by injecting messages into an ongoing CPDLC session. These attacks can have severe consequences, as CPDLC messages can be hijacked and tampered with to change critical instructions like declaring emergencies or altering aircraft altitudes and speeds. The feasibility of these attacks was proven by launching them from a location hundreds of kilometres away.

II. Our Contributions

While 5G handover achieves some security properties, the attacks discussed here illustrate that these current handover deployments lack a cohesive security framework, and that some attacks exploit distinct functionalities of handovers: consider the CPDLC attack that triggers a false handover, or the ability to hijack sessions during the handover phases. Formalising distinct notions of security for handover schemes clarifies security guarantees necessary during a handover phase of a protocol, preventing the possibility of previously demonstrated attacks [7]. We emphasise that previous work that model handovers as a variation of the *key exchange* primitive [8], [9] which do not appropriately capture handover protocols.

We generically define a handover scheme HO as a cryptographic protocol executed between three parties: a user U, a source S, and a target T. U is mobile, travelling 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 U's transition between these zones: specifically, we assume that U has previously communicated with the current station in their zone (which we denote the source S), and wishes to continue their current communication session in the proceeding zone with the new station, target T. A handover scheme allows S to communicate and authenticate sufficient information to T, allowing U to continue their communication session with T without re-executing a full handshake between U and T - usually by establishing a shared secret key.

a) 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. For example, a three-party key exchange protocol would require all parties jointly establish shared secret keys 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* know 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 a HO, there exists an asymmetry to the pre-established trust relationships. For example, neighbouring stations are likely to know each other's public keys, but T is unlikely to know a mobile user's U public keys. Additionally, S and U are likely to share pre-established symmetric keys due to some previous handover. Thus, S 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.

Finally, HOs should prepare for the next HO: while multi-stage key exchanges [10], [11] output keys over many rounds, these are continually executed with the same partner, whereas the transition to new partners is core to both HO protocols and our new path privacy notion.

The pre-establishment of keys, the asymmetric trust relationships and the continuity-preserving transition of communication collectively set HO apart from key exchange protocols as a unique primitive. The widely-adopted 5G handover protocol [1] exemplifies all these characteristics that we have identified as unique to HO. Thus, we endeavour to address this gap and systemically treat HO, formalising it as a distinct primitive and capture its security.

b) Path Privacy: On a high-level, path privacy captures the notion that S and T nodes should not learn each other's identities. Multiple instances of subscriber privacy violations and data breaches by cellular network operators have drawn significant attention in recent years. For example, the Federal Communications Commission recently fined AT&T, Sprint, T-Mobile, and Verizon millions for unlawfully sharing users' geolocation data with third parties within their commercial programs, including prisons and bounty hunters [12], [13]. In one case, a bounty hunter [13] successfully tracked their target to a specific neighbourhood in New York, just blocks away from their actual location.

Similarly, the growing trend of manufacturing increasingly smart vehicles has heightened privacy concerns for their owners. Privacy researchers at the Mozilla Foundation revealed that smart car manufacturers often collect excessive personal data, with 17 out of the 25 manufacturers reviewed actively selling the data they gather [14]. A recent article published by WIRED [15] highlighted security vulnerabilities in a Subaru employee web portal that allowed access to comprehensive vehicle location data. This included the car's precise location each time its engine started, with records spanning up to an entire year. These privacy violations are further exacerbated by the inherent design of smart vehicular infrastructures. Components such as roadside units (RSUs), which provide services to smart vehicles, often lack privacy safeguards and can be exploited to track vehicle locations [16].

In contrast, commercial aircraft travel plans are often public and available days in advance. However, military, government, and private business stakeholders require greater location privacy. For example, the movements and communications of

a business flight may need to be kept private due to their potential impact on stock prices (e.g., mergers, acquisitions) [17]. 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, given the use of open and unencrypted channels in avionic communication, it may yield these attempts at privacy futile [18] revealing sensitive information such as location data. In fact, the work of Strohmeier et al. [19] 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, failing to conceal a user's footprint can leak information about their physical location.

Prior work that investigate privacy issues within existing and future cellular architecture [20]–[22] have proposed solutions ranging from global identifier with anonymous authentication [22] to virtual private mobile networks (VPMNs) comprising mutually-trusted device-to-device communication [21], [22]. However, these solutions do not leverage the inherent properties of handovers in their constructions nor do they formally analyse the security of their proposed schemes. For example, the property of identity privacy discussed in [22] conflates the distinctions between anonymity and unlinkability. In Section IV we identify these two properties as separate and argue that unlinkability provides stronger security guarantees. Furthermore, our work introduces *path privacy*, offering a simpler and practical framework that integrates privacy guarantees into existing infrastructure, backed by formal security proofs.

c) Formalised Security Notions & Capturing Stronger Security: 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 indistinguishability* 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 [8], [9]. We implicitly capture *mutual authentication* as a core security property, to eliminate the threat of session-hijacking attacks against HO [7]. In Section V we propose a generic HO scheme that achieves our formalised security goals, demonstrating how to realise strong notions of security in a HO setting. In our construction in Figure 6, 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*.

Summary of Contributions. Thus, in this work we introduce a universal formalised framework that captures HO schemes as a unique primitive; 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; introduce a generic strong HO scheme that achieves our defined path privacy property alongside other security goals with formal analysis of its security within our framework; additionally in Appendix A and B we investigates the general applicability of our formalisation by mapping 5G-HO protocol to our framework, and illustrate its flexibility to model beyond 5G, respectively.

Organisation. We begin by evaluating existing literature in Section III. In Section IV, we formalise our notions of secure handover as a primitive. We further detail achievable security goals for secure handovers along with their respective models that capture each identified security goal. Section V we present a generic strong HO scheme constructed within our framework that achieves all our identified notions of security. In Section VI, we analyse the security of our protocol we introduced in Section V. The paper closes with conclusions and directions for future work in Section VII.

III. 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 [8], [9], [23]–[28], schemes have been proposed for HO within cloud computing architectures [29], urban air mobility (UAM) networks [3] and VANETs [30].

We list all the works studied in Table I, describing claimed security properties and the status of their formal security analysis. While *unlinkability* and *anonymity* are often used interchangeably, we define them as distinct security properties. *Unlinkability* 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 *unlinkability* is compromised. However, maintaining *unlinkability* ensures *anonymity*, as it prevents an adversary from linking sessions to an individual user. Thus, we treat *unlinkability* and *anonymity* as distinct properties, with *unlinkability* providing stronger security guarantees.

Provable Security of Secure Handovers. A diverse selection of methodologies have been utilised to analyse the security of secure handover protocols, and some work that introduces novel handover schemes do not formally analyse the protocols at all [29]. Indeed, while most works surveyed claim anonymity, unlinkability, only the works of [8] and [9] provide formal proofs to verify these security notions. However, their work is exclusively modelled after the 5G-HO protocol [1] 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. [1] formally analyse the security of the 5G-HO scheme in the symbolic model, using the verification tool Tamarin. Alnashwan et al. [8] and Fan et al. [9] propose security improvements for the standard 5G-HO protocol and analyse the security of their proposed schemes in the computational model. Norrman [4] models 5G-HO as a secure anycast channel [33] and comes closest to our work, but is strongly tied to 5G-HO specifically, with assumptions of pre-existing secure-channels for communication and a central

Properties / Work	PFS	Key Indistinguishability	Unlinkability	Mutual Authentication	Anonymity	Source Privacy	Target Privacy
[30]	●[AVISPA]	●[CP]	○	●[BAN/AVISPA]	○	○	○
[23]	●[Scyther]	○	○	●[BAN/Scyther]	○	○	○
[24]	○	○	○	○[BAN]	○	○	○
[25]	○	○	○	○[BAN]	○	○	○
[9]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	○	○
[8]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	○	○
[26]	●[AVISPA]	●[AVISPA]	○	●[BAN/AVISPA]	○	○	○
[27]	●[Scyther]	●[Scyther]	○	●[BAN/Scyther]	○	○	○
[29]	○	○	○	○	○	○	○
[28]	●[CP/AVISPA]	●[CP/AVISPA]	○	●[CP/AVISPA]	○	○	○
[3]	●[AVISPA]	●[CP]	○	●[BAN/AVISPA]	○	○	○
[31]	●[Tamarin]	○	○	●[Tamarin]	○	○	○
[32]	○	○	○	○	○	○	○
Our Work	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]	●[CP]

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

orchestrator entities for service provision, which does not fit other HO constructions such as CPDLC. Moreover, their work does not capture *user unlinkability* and *path privacy* properties and proves confidentiality and integrity of data transmissions.

A significant number of examined works applies BAN-logic [34] to prove the properties of mutual authentication and key agreement in their proposed schemes. However, the suitability of BAN-logic as a framework to analyse the security of protocols has been contested and the works of [35], [36], and [37] capture serious security flaws in protocols proven secure under the BAN-logic. As such, in our study we consider work that solely analyse the security of their proposed schemes with BAN-logic as insufficient, and have categorised them as work providing no formal security proofs in Table I. Conversely, the works of [3], [23], [26], [27], [30] 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 Figure 6 is the first to achieve all properties simultaneously. We now turn to introducing our formalism for secure handover schemes.

IV. Secure Handover Formalisation and Security

Here 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 experiment that captures each goal.

A. Formalising Handovers

We consider a protocol that is executed between three parties: a user U, a source S and a target T. User 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. We limit our formalisation to three parties since 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 generalises 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); the U and the S generate some shared secret state, a bootstrap key bk[1] to enable the handover, and agree on some additional data ad that needs to be advocated to the T (abstractly capturing an initial key exchange, or a previous handover);
- A *preparation* phase, where the U and S interact to generate some material that allows the S to authenticate information that will be used by the user to communicate to the T. This preparation phase allows for the handover protocol to achieve *source* or *target privacy* by bypassing a need for S and T to communicate directly;
- A *support* phase, where the S and the 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 the 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 $\mathsf{HO} = \{\mathsf{Gen}, \mathsf{SGen}, \mathsf{Setup}, \mathsf{Prep}, \mathsf{Supp}, \mathsf{Cont}\}$:

- $\mathsf{Gen}(1^\lambda) \xrightarrow{\$} (pk, sk, \mathsf{pid})$: 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 ($tpid$).
- $\mathsf{SGen}(1^\lambda) \xrightarrow{\$} (bk, \mathsf{id})$: 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 allows user U to leverage 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 (potentially) 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 facilitates session management per protocol execution by explicitly detailing states maintained between parties.

[1]The 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.

– $\mathsf{Prep}(st, pk_{\mathtt{T}}, m) \xrightarrow{\$} (st', m)$: is a probabilistic 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 realisation 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 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. The deletion of relevant state data by source S in this manner is essential to prevent an adversary, that compromises S after the fact, from recovering the same secrets or (if impersonating the target T) breaking key indistinguishability.
– $\mathsf{Cont}(st, pk_j, pk_{\mathtt{S}}, m) \xrightarrow{\$} (st', m')$: is a probabilistic algorithm run by U and T, which takes as input the secret state st (containing session keys k and handover keys hk), the long-term public key of the communicating party pk_j (where $j \in \{\mathtt{U}, \mathtt{T}\}$), the long-term public key of 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 2.

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 formalisation can be added or subtracted according to the demands of the specific design. For instance, in our strong HO scheme proposed in Section V, we forgo the Supp phase in order to capture the property of *path privacy* in our construction. The proposed secure LDACS-HO [31] does not include a Contact phase since S acts as an intermediary throughout, forwarding messages between U and T and no direct communication takes place between U and T until the HO is completed. Our framework therefore provides a highly customisable and flexible structure that formalises aspects of network limitations as seen with LDACS [31], while also encouraging the integration of stronger security notions, which we demonstrate in our StrongHO construction in Section V. In Figure 9, we illustrate the flexibility of our framework by mapping the existing 5G [1] and CPDLC HO [7] protocols and the proposed LDACS-HO [31] to our construction. Moreover, in Appendix A we further investigate the universal applicability of our formalisation by capturing the 5G-HO protocol within our formalised framework.

B. 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 U and the target T by interacting with the source 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 generalisation of our approach, we define *key indistinguishability* of session keys established between U and T as the primary goal of secure handover schemes. Since our formalism also produces handover keys, established between U and 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 (k, hk) derived in the protocol execution, or random keys from the same distribution. $\mathcal{A}$'s goal is to determine which keys they have been given.

We formalise this goal in Figure 3. We describe below the per-session variables maintained by each session instance. Next, we give the explicit definition of security below and state $\mathcal{A}$'s advantage in winning this game.

Execution Environment: Here we describe the execution environment of all experiments for the proposed scheme. 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.
– $T_{\mathtt{P}}, T_{\mathtt{S}}, T_{\mathtt{H}}$: Session transcripts of the Prep, Supp and Cont algorithms respectively, initialised by $\perp$.
– $\alpha \in \{\mathtt{prep}, \mathtt{supp}, \mathtt{contact}, \mathtt{accept}, \mathtt{reject}, \perp\}$: The current status of the session, initialised with $\perp$.
– $bk \in \{\{0,1\}^\lambda, \perp\}$: Bootstrap key used as the initial shared secret between S and U, or $\perp$.
– $k \in \{\{0,1\}^\lambda, \perp\}$: Session key to be used in some following symmetric key protocol, initialised as $\perp$.
– $hk \in \{\{0,1\}^\lambda, \perp\}$: Handover key used as bk in some following handover, initialised as $\perp$.
– $st \in \{0,1\}^\lambda$: 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 $\mathcal{A}$ to create a new session π with role ρ owned by party i, with communicating

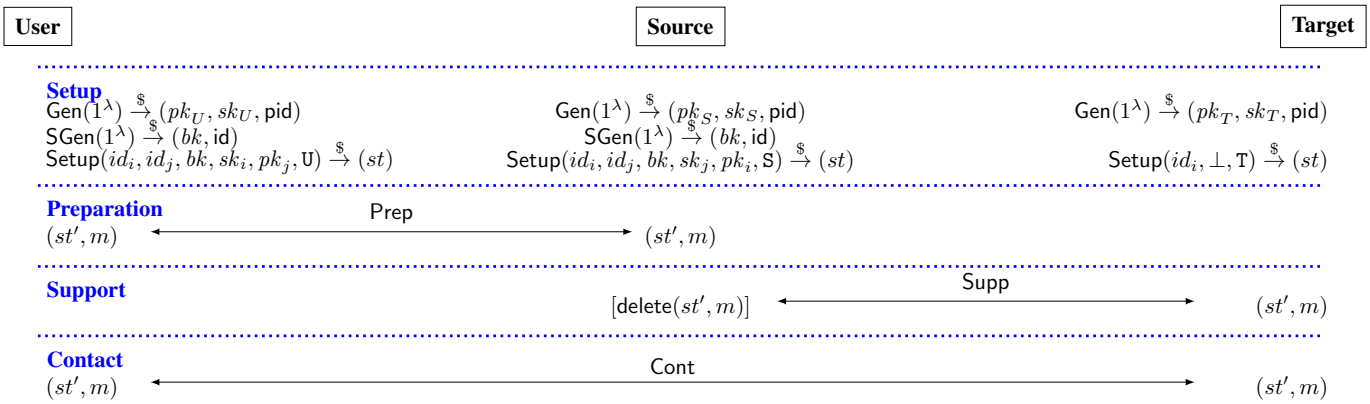

Fig. 2: An expected execution of a secure HO protocol.

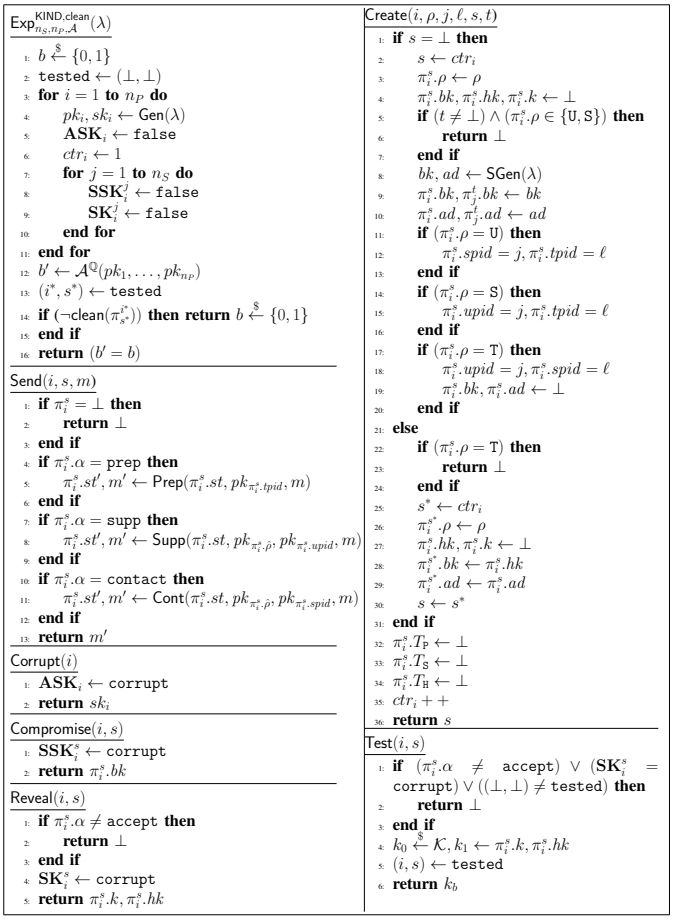

Fig. 3: The key indistinguishability security experiment for secure HO schemes. For conciseness we use $\pi_i^s.\hat{\rho}$ as shorthand for the communicating partner's party index, i.e. for Prep $\pi_i^s.\hat{\rho} = \pi_i^s.spid$ if $\pi_i^s.\rho = \mathtt{U}$, and $\pi_i^s.upid$ otherwise; for Supp $\pi_i^s.\hat{\rho} = \pi_i^s.tpid$ if $\pi_i^s.\rho = \mathtt{S}$ and $\pi_i^s.spid$. $\mathbb{Q}$ denotes the set of all queries used in the experiment, i.e. $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{Test}\}$.

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 handover, that it is done consistently, aborting if not.

- Send(i, s, m): allows $\mathcal{A}$ 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 $\mathcal{A}$.
- Corrupt(i): allows $\mathcal{A}$ to recover the long-term secrets of party i, which enables the framework to capture perfect forward secrecy.
- Compromise(i, s): allows $\mathcal{A}$ to recover the bootstrap key bk used by π_i^s in their protocol execution.
- Reveal(i, s): allows $\mathcal{A}$ to reveal the session and handover key computed by π_i^s in their protocol execution, allowing our model to capture key independence.
- Test(i, s): returns to $\mathcal{A}$ the real-or-random session key and handover keys computed by the *test session* π_i^s, allowing $\mathcal{A}$ to play the key indistinguishability game. This query can only be called once.

Cleanness Predicate: Our adversary can use the Corrupt, Compromise and Reveal queries to learn secrets, and can trivially impersonate the exposed party to their communicating partner, thus learning the secrets of a potential test session π_i^s. To prevent trivial attacks, we define *cleanness predicates* that prevent the adversary from making particular patterns of adversarial queries relative to π_i^s. 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 protocol has long-term asymmetric authentication secrets. Below we define a cleanness predicate for our StrongHO protocol described in Figure 6.

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. $T_{\mathtt{P}}$ for the Prep execution, $T_{\mathtt{S}}$ for the Supp execution and $T_{\mathtt{H}}$ 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 $\pi_i^s.T_{\mathtt{H}} = \pi_j^t.T_{\mathtt{H}}$.

Definition 1 (Strong Handover KIND cleanness predicate)**.** *A session π_i^s such that $\pi_i^s.\alpha = \mathtt{accept}$ in the security experiment defined in Figure 3 is clean (i.e., $\mathsf{clean}_{\mathsf{str\text{-}kind}}(\pi_i^s) = 1$) if all of the following conditions hold:*

1) $\mathbf{SK}_i^s \neq \mathtt{corrupt}$ ***(Session key has not been exposed)****;*
2) $\forall (j,t) \in n_P \times n_S$ *such that π_i^s UT-matches π_j^t,* $\mathbf{SK}_j^t \neq \mathtt{corrupt}$ ***(Session key not exposed at partner session)****;*
3) *If $\pi_i^s.\rho = \mathtt{U}$, 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* $\mathbf{SSK}_i^s \neq \mathtt{corrupt}$ *nor* $\mathbf{SSK}_l^r \neq \mathtt{corrupt}$ ***(If the user session has no matching*** UT ***partner, then their bootstrap key has not been exposed)****;*
4) *If $\pi_i^s.\rho = \mathtt{U}$, 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* $\mathbf{SK}_l^r \neq \mathtt{corrupt}$ ***(If the user session has no matching*** UT ***partner, then their previous handover key has not been exposed)****;*
5) *If $\pi_i^s.\rho = \mathtt{T}$, 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* $\mathbf{SK}_l^r \neq \mathtt{corrupt}$ ***(If the target session has no matching*** UT ***partner, then the user partner's bootstrap key has not been exposed)****;*
6) *If $\pi_i^s.\rho = \mathtt{T}$, 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* $\mathbf{SK}_l^r \neq \mathtt{corrupt}$ ***(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* $\mathbf{ASK}_i \neq \mathtt{corrupt}\ \forall\ i$ ***(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 $\pi_i^s.\rho = \mathtt{U}$ then* $\mathbf{ASK}_{\pi_i^s.tpid} \neq \mathtt{corrupt}$ ***(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 2 (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 defined in Figure 3 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.*

C. 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. To capture this, we formalise 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.

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 interacting with $\mathcal{A}$. 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 formalise this goal in Figure 4.

As before, 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: all other queries remain identical.

- $\mathsf{TestUnlink}((i,s),(i',s'),j,(t,t'),l)$: 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.
- $\mathsf{SendTest}(m)$: allows the adversary to send a message m to the Test session π_b.

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.

Definition 3 (Strong Handover UNLINK cleanness predicate)**.** *A session π_i^s such that $\pi_i^s.\alpha = \mathtt{accept}$ in the security experiment defined in Figure 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 1)***
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 1)***

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

Definition 4 (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.

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

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** $(\neg\mathsf{clean}(\pi_b) \vee \neg\mathsf{clean}(\pi_{b-1}))$ **then**
13: $\quad$ **return** $b \xleftarrow{\$} \{0,1\}$
14: **else**
15: $\quad$ **return** $(b' = b)$
16: **end if**

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

1: $\mathsf{Send}(\pi_b, m) \to m'$
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** $(\mathbf{SK}^s_i = \mathtt{corrupt}) \vee (\mathbf{SK}^{s'}_{i'} = \mathtt{corrupt}) \vee (\mathbf{SK}^t_j = \mathtt{corrupt}) \vee (\mathbf{SK}^{t'}_{j'} = \mathtt{corrupt})$ **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: $\quad \pi_b \leftarrow \pi^s_i$
19: $\quad \pi_{b-1} \leftarrow \pi^{s'}_{i'}$
20: $\quad r \leftarrow \mathsf{Create}(\ell, \mathtt{T}, i, j, \perp)$
21: $\quad \pi^s_i, \pi^{s'}_{i'}, \pi^{t'}_j \leftarrow \perp$
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: $\quad \pi_{b-1} \leftarrow \pi^s_i$
26: $\quad \pi_b \leftarrow \pi^{s'}_i$
27: $\quad r \leftarrow \mathsf{Create}(\ell, \mathtt{T}, i', j, \perp)$
28: $\quad \pi^t_j \leftarrow \pi^{t'}_j$
29: $\quad \pi^s_i, \pi^{s'}_{i'}, \pi^{t'}_j \leftarrow \perp$
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)

Fig. 4: The unlinkability security experiment for secure handover 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. $\mathbb{Q}$ denotes the set of all queries used in the experiment, i.e. $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{TestUnlink}, \mathsf{SendTest}\}$.

D. 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) is private information that is worth protecting. For instance, in 5G identifying the base stations that the user communicates with would enable an attacker to recover their general geographical location. To formalise the security of this information in our framework, we introduce Target and Source Privacy (which we denote TPRIV and SPRIV respectively) which we collectively identify as *path privacy*. 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) owns the Test session interacting with $\mathcal{A}$.

However, 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 formalise this game in Figure 5. We give the explicit definition of security below and state $\mathcal{A}$'s advantage in winning this game.

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

1: $b \xleftarrow{\$} \{0,1\}$
2: **for** $i = 1$ **to** n_P **do**
3: $\quad pk_i, sk_i \leftarrow \mathsf{Gen}$
4: $\quad \mathbf{ASK}_i \leftarrow \mathtt{false}$
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** $(\neg\mathsf{clean}(\pi_b) \vee \neg\mathsf{clean}(\pi_{b-1}))$ **then**
13: $\quad$ **return** $b \xleftarrow{\$} \{0,1\}$
14: **else**
15: $\quad$ **return** $(b' = b)$
16: **end if**

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

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

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

1: **if** $(\mathbf{ASK}_\ell = \mathtt{corrupt}) \vee (\mathbf{ASK}_{\ell'} = \mathtt{corrupt})$ **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** $(s' = \perp) \vee (t' = \perp)$ **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)

Fig. 5: The target privacy security experiment for secure handover 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. $\mathbb{Q}$ denotes the set of all queries used in the experiment, i.e. $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{TestTarget}, \mathsf{SendTest}\}$.

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. 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.

Definition 5 (Strong Handover TPRIV cleanness predicate)**.** *A session* π^s_i *such that* $\pi^s_i.\alpha = \mathtt{accept}$ *in the security experiment defined in Figure 5 is* $\mathsf{clean}_{\mathsf{str\text{-}tpriv}}$ *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 UT*-matches* π_j^t, $\mathbf{SK}_j^t \neq \texttt{corrupt}$ ***(Session key not exposed at partner session)***;
3) $\mathbf{ASK}_i \neq \texttt{corrupt}$ ***(Target's long-term key has not been exposed)***;
4) *If 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)***;
5) *If 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)***;

Definition 6 (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:* $\mathsf{Adv}^{\mathsf{TPRIV},\mathsf{clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{TPRIV},\mathsf{clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$.

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, we give a similar definition of security for source privacy in Appendix D. We also give a formalisation of the source privacy game in Figure 10. The cleanness predicate for SPRIV is provided in Appendix E. We finish by giving a formal definition for path privacy, which encapsulates both SPRIV and TPRIV.

Definition 7 (Path Privacy for Handover Schemes)**.** *Let* HO *be a secure handover protocol, and* n_P*,* $n_S \in \mathbb{N}$*. For the advantages for* SPRIV *(Definition 8) and* TPRIV *(Definition 6), and a PPT algorithm* $\mathcal{A}$*, we define the advantage of* $\mathcal{A}$ *against path privacy to be:* $\mathsf{Adv}^{\mathsf{PPRIV}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) = \mathsf{Adv}^{\mathsf{TPRIV},\mathsf{clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda) + \mathsf{Adv}^{\mathsf{SPRIV},\mathsf{clean}}_{\mathsf{HO},n_P,n_S,\mathcal{A}}(\lambda)$*. 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*.*

V. Strong Handover Scheme

In this section, we construct a generic strong handover protocol that captures all notions of security that we describe in Section IV. We note that our construction is not intended as a drop-in replacement for a specific handover scheme, but a generic construction that achieves the strongest degree of security, which can be downgraded as necessary for a given setting. For instance, path privacy is too strong for commercial aviation where the trajectory of an aircraft is regularly tracked for safety reasons. However, users transitioning between regions in 5G undoubtedly benefit from the additional location-privacy guarantees of path privacy.

In our construction, we present a strong handover protocol that captures all previously formalised security notions within the handover setting (KIND, UNLINK), including our novel path privacy notions (SPRIV, TPRIV). On a high-level, the user U generates a ciphertext $ctxt_{\mathtt{T}}$ (encrypting a symmetric key under the target T's public key) and a KEM public key $epk_{\mathtt{U}}$. Both are passed to the source S for authentication, which signs and MACs both values. U verifies and deletes the signature, sending the KEM public key $epk_{\mathtt{U}}$, the ciphertext $ctxt_{\mathtt{T}}$ and the MAC tag to T. T verifies the MAC tag and is satisfied that both values come from some advocated-for U. T encapsulates a fresh secret under U's public key $epk_{\mathtt{U}}$, and both parties derive the same set of keys k, hk.

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. The PPRF fortifies the security of our construction by puncturing ak after it anonymously authenticates S to T for the ongoing HO session, preventing replay attacks and guaranteeing forward secrecy. StrongHO has only three phases, which are described below and illustrated in Figure 6.

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 $ctxt_{\mathtt{T}}$. 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 path privacy, 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 verifies 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}. 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.

Trade-offs between Path Privacy and Compromise Resilience: One observation we made during our design is that there seems to exist 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 similarly to our PPRF secret. 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 seems clear that this trade-off is inherent to these properties: the larger the group that a source S is indistinguishable from, the larger the set of source nodes an attacker can compromise to forge messages from S.

In our scheme illustrated in Figure 6, we circumnavigate this trade-off by exploiting U's role in authenticating S, at the expense of computational efficiency. 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 strong HO 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*.

VI. Security Analysis

In this section we provide an analysis of the secure HO protocol that we introduced in Section V. In particular, we provide a comprehensive proof sketch for key indistinguishability of the StrongHO protocol to demonstrate how the analysis of a security property occurs in our framework. We refer the reader to Appendix C for detailed proofs for KIND security of StrongHO. Next we provide proof sketches of all other properties of our StrongHO protocol.

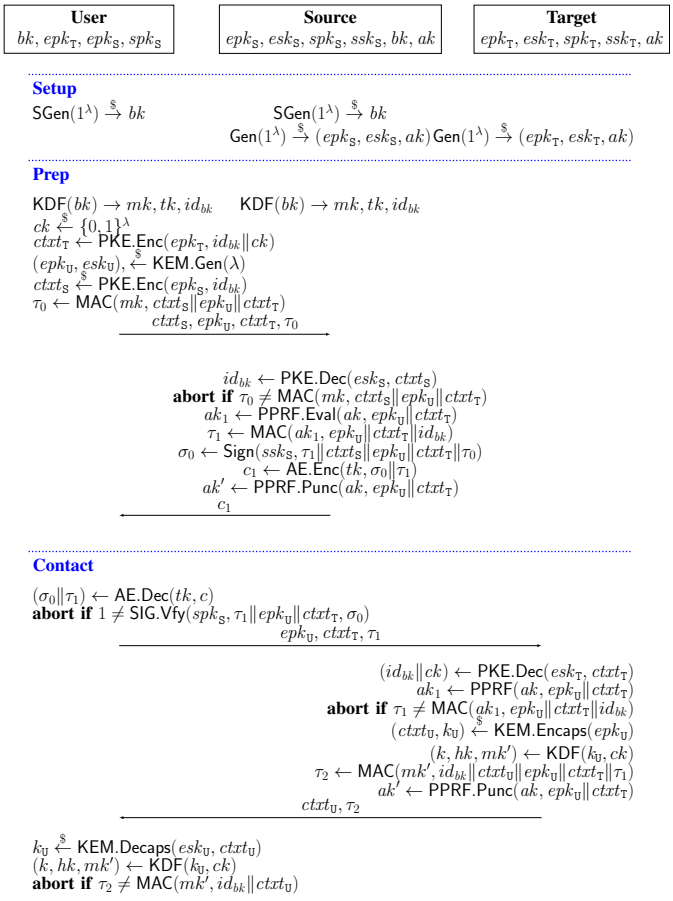

Fig. 6: The StrongHO protocol.

We begin with proving the KIND security of the StrongHO protocol, described in Figure 6. The cryptographic assumptions that we use can be found in Appendix F.

Theorem 1 (StrongHO KIND Security)**.** *The* StrongHO *protocol presented in Figure 6 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)* $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)$ *is negligible under the* prf, pprf, eufcma, ind-cpa, ind-cca, ikcca *and* eufcma *security of the* PRF, PPRF, MAC, KEM, PKE, PKE *and* SIG *primitives respectively.*

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.

In what follows we define $\mathcal{A}$'s advantage in Case X as $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_X}(\lambda)$. 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. 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 end of this case we show that there exists an honest session π^t_j such that $ctxt_{\mathtt{U}} \in \pi^s_i.T_{\mathtt{H}}, ctxt'_{\mathtt{U}} \in \pi^t_j.T_{\mathtt{H}}, ctxt_{\mathtt{U}} = ctxt'_{\mathtt{U}}$, and $epk_{\mathtt{U}} \in \pi^s_i.T_{\mathtt{H}}, epk'_{\mathtt{U}} \in \pi^t_j.T_{\mathtt{H}}, epk_{\mathtt{U}} = epk'_{\mathtt{U}}$.

Game 0 is the initial KIND security game. In **Game 1** and **Game 2**, we guess the index (i, s) of the session π^s_i, and the index (j, t) of the source partner π^t_j respectively, incurring a tightness loss of $n_P n_S$ each. In **Game 3**, we introduce an abort event $\mathtt{event_S}$ that occurs if the Test session π^s_i sets $\alpha = \mathtt{accept}$ without an honest US-match and in the following games, we bound this advantage. In **Game 4** 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 $\mathcal{B}_1$ to a PRF assumption. In **Game 5** $\mathcal{C}$ 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 $\widetilde{tk}$ via a reduction $\mathcal{B}_2$ to an AEAD auth challenger $\mathcal{C}_{\mathsf{auth}}$. Whenever $\mathcal{C}$ is required to encrypt/decrypt using $\widetilde{tk}$, $\mathcal{B}_2$ instead queries the ciphertext/plaintext $\mathcal{C}_{\mathsf{auth}}$. By **Game 4** $\widetilde{tk}$ is uniformly random and independent, 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 13. We note that the ciphertext c_1 contains (and thus authenticates) the signature σ_0, which itself is computed over all messages received by S from the U. Thus, U now aborts if they complete the preparation phase without a US-matching partner, and $\Pr(\mathtt{event}_\alpha) = 0$. In **Game 6**, $\mathcal{C}$ guesses the party index ℓ of the intended target partner of the test session π^s_i, and in **Game 7** $\mathcal{C}$ introduces an abort event $\mathtt{event_T}$ that occurs if the Test session π^s_i sets $\alpha = \mathtt{accept}$ without an honest UT-match. We note that by definition of the case, π^s_i never has a UT-match and thus $\mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda) = 0$, since $\mathcal{A}$ can never test a session that aborts before $\alpha \leftarrow \mathtt{accept}$. In what follows, we bound $\Pr(\mathtt{event_T})$. In **Game 8**, $\mathcal{C}$ replaces ck in $ctxt_{\mathtt{T}}$ computed by π^s_i with a random string of the same length $\widetilde{ck}$. We construct a reduction $\mathcal{B}_3$ that interacts with an ikcca 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 $\mathcal{C}$, $\mathcal{B}_3$ initialises a ikcca challenger $\mathcal{C}_{\mathsf{ikcca}}$, and replaces pk_l with pk output by $\mathcal{C}_{\mathsf{ikcca}}$. When π^s_i computes $ctxt_{\mathtt{T}}$, $\mathcal{B}_3$ instead picks a uniformly random binary string z' of length equal to $z = id_{bk} \| ck$ and submits (z, z') to the PKE.Enc oracle. For any decryption operations requiring sk_ℓ, $\mathcal{B}_3$ submits the query to its respective Dec oracle, except for decrypting $ctxt_{\mathtt{T}}$, where it simply sets the output to z. By definition of cleanness condition 8 of $\mathsf{clean}_{\mathsf{str\text{-}kind}}$ and Case 1, $\mathcal{A}$ cannot issue $\mathsf{Corrupt}(\ell)$. Thus, $\mathcal{A}$ cannot know any information about ck, since it is never communicated to $\mathcal{A}$. **Game 9** is identical to **Game 4**, where $\mathcal{C}$ replaces the derived keys $k, hk, mk' = \mathsf{KDF}(k_{\mathtt{U}}, \widetilde{ck})$ with uniformly random values $\widetilde{k}, \widetilde{hk}, \widetilde{mk'} \xleftarrow{\$} \{0,1\}^{\mathsf{PRF}}$ by interacting with a PRF challenger. In **Game 10** $\mathcal{C}$ aborts if $\mathcal{A}$ is able to successfully forge τ_2 to the Test session, by interacting with a MAC challenger $\mathcal{C}_{\mathsf{eufcma}}$. Note that the MAC tag authenticates all messages sent in the Contact phase, so by **Game 10** π^s_i now aborts before accepting without a 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 $\pi^s_i.\rho = \mathtt{T}$. **Game 0** is the standard KIND security game and in **Game 1** we guess the index (i, s) of the target session π^s_i at a tightness loss of $n_S n_P$. In **Game 2**, we replace the computation of ak_1 by π^s_i 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 $\mathcal{B}_6$ initialises 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. 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 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 $\mathsf{PPRF}.\mathcal{C}(epk_{\mathtt{U}} \| ctxt_{\mathtt{T}})$, 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 11. In **Game 3** $\mathcal{C}$ aborts if $\mathcal{A}$ is able to successfully forge τ_1 to the Test session, by interacting with a MAC challenger $\mathcal{C}_{\mathsf{eufcma}}$ and in **Game 4** we guess the honest source session π^k_u that produced the MAC tag τ_1 at a tightness loss of $n_P n_S$. In **Game 5** $\mathcal{C}$ 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^k_u (and its corresponding user session) by interacting with a PRF challenger as in **Case 1 Game 4**. In **Game 6**, $\mathcal{C}$ aborts if $\mathcal{A}$ is able to successfully forge τ_0 to the guessed source session π^k_u, by interacting with a MAC challenger $\mathcal{C}_{\mathsf{eufcma}}$ as in **Case 1, Game 10**. By **Game 6** we know that there exists an honest user session that communicated with π^k_u without modification. This π^k_u produced τ_1 honestly, which authenticates the public key $epk_{\mathtt{U}}$ and ciphertext $ctxt_{\mathtt{T}}$ received by the target session π^s_i. Thus, by **Game 6** we have that there exists some honest user session that UT-matches π^s_i, 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 is the initial KIND security game and in **Game 1**, $\mathcal{C}$ guesses the index of the test session (i, s) and its UT matching

partner π_j^t at a tightness loss of ${n_P}^2 {n_S}^2$. In **Game 2**, $\mathcal{C}$ 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-cpa 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-cpa KEM challenger, and the computation of $k_{\mathtt{U}}$ with the output key. Detecting the replacement of $k_{\mathtt{U}}$ implies an efficient distinguishing PPT algorithm $\mathcal{A}$ against ind-cpa security of KEM. In **Game 3**, $\mathcal{C}$ 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 interacting with a PRF challenger as in **Game 5** of **Case 2**. Here we emphasise 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 $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) = 0$.

Theorem 2 (StrongHO UNLINK Security)**.** *The* StrongHO *protocol presented in Figure 6 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 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*,* eufcma*,* ind-cpa*,* ind-cca*,* ikcca *and* eufcma *security of the* PRF*,* MAC*,* KEM*,* PKE*,* PKE *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_i^s - 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_i^s 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_i^s 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 bounds of the KIND proof of StrongHO, StrongHO achieves UNLINK security.

Theorem 3 (StrongHO SPRIV Security)**.** *The* StrongHO *protocol presented in Figure 6 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 10)* $\mathsf{Adv}^{\mathsf{SPRIV},\mathsf{clean}_{\mathsf{str\text{-}spriv}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S}(\lambda)$ *is negligible under the* prf*,* pprf*,* eufcma*,* ind-cpa*,* ind-cca*,* ikcca *and* eufcma *security of the* PRF*,* PPRF*,* MAC*,* KEM*,* PKE*,* PKE *and* SIG *primitives.*

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 S. The signature σ_0, however is signed using the public long-term signing key $spk_{\mathtt{S}}$ of S, which could reveal the identity of the S to a potential adversary. However, we encrypt σ_0 along with τ_1, and thus only the U who shares tk with the source learns the identity of 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 initialising 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 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.

Theorem 4 (StrongHO TPRIV Security)**.** *The* StrongHO *protocol presented in Figure 6 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 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*,* eufcma*,* ind-cpa*,* ind-cca*,* ikcca *and* eufcma *security of the* PRF*,* PPRF*,* MAC*,* KEM*,* PKE*,* PKE *and* SIG *primitives.*

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, 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 6 is* PPRIV*-secure.*

Proof. *The proof follows from Theorem 3 and Theorem 4.*

VII. Conclusion and Remarks

In this paper we have presented a formalisation framework for secure handovers recognising its uniqueness as a distinct primitive. We leverage our formalisation to propose *path privacy*, a stronger notion of privacy against insider attacker and, we proceed to capture this notion in our proposed StrongHO scheme. We further highlight that the distinctness of handovers as a primitive is fundamental to capturing our proposed property of *path privacy*. Other comparable primitives such as key exchanges are not designed to facilitate secure transition of an existing session from one party to another and thus are ill-fitted to integrate *path privacy* within their constructions. By recognising handovers as a distinct and standalone primitive, we were able to identify unique security challenges inherent to its construction and propose suitable security properties.

We note that our HO formalism only implicitly models *trusted* nodes, such as those represented by 5G home networks, which may limit its applicability in some contexts. Furthermore, our HO model assumes *honest-but-curious* 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.

Nevertheless, the modularity and flexibility of our framework opens many new avenues for research. Our formalisation can be leveraged to analyse the security of known HO constructions such as 5G-HO and the proposed secure LDACS-HO for aviation, as mapped in Figure 9. Other constructions that could be considered as handover schemes in our framework include OAuth and eduroam, and applying our framework to these may highlight unknown security properties.

References

[1] 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*, pp. 1–12, 2021.

[2] ICAO, "Global operational data link document (gold) manual," pp. 24–34, International Civil Aviation Organization, 2013.

[3] 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*, vol. 10, pp. 42529–42541, 2022.

[4] K. Norrman, "Secure anycast channels with applications to 4g and 5g handovers," *Cryptology ePrint Archive*, 2022.

[5] 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)*, pp. 369–374, IEEE, 2018.

[6] 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*, pp. 1383–1396, 2018.

[7] 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*, pp. 53–64, 2021.

[8] 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*, vol. 18, pp. 1898–1913, 2023.

[9] 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*, vol. 15, pp. 927–942, 2019.

[10] M. Fischlin and F. Günther, "Multi-stage key exchange and the case of google's quic protocol," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1193–1204, 2014.

[11] B. Dowling, M. Fischlin, F. Günther, and D. Stebila, "A cryptographic analysis of the tls 1.3 handshake protocol," *Journal of Cryptology*, vol. 34, no. 4, p. 37, 2021.

[12] B. Fung, "Fcc fines wireless carriers millions for sharing user locations without consent," *CNN*, Apr. 2024.

[13] 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.

[14] J. Caltrider, M. Rykov, and Z. MacDonald, "It's official: Cars are the worst product category we have ever reviewed for privacy," 2023.

[15] A. Greenberg, "Subaru security flaws exposed its system for tracking millions of cars," 01 2025.

[16] T. Yoshizawa, D. Singelée, J. T. Muehlberg, S. Delbruel, A. Taherkordi, D. Hughes, and B. Preneel, "A survey of security and privacy issues in v2x communication systems," *ACM Comput. Surv.*, vol. 55, Jan. 2023.

[17] T. Maremont, Mark McGinty, "Ready for Departure: M&A Airlines," *The Wall Street Journal*, 2011.

[18] 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.

[19] 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.

[20] S. Tomasin, M. Centenaro, G. Seco-Granados, S. Roth, and A. Sezgin, "Location-privacy leakage and integrated solutions for 5g cellular networks and beyond," *Sensors*, vol. 21, no. 15, p. 5176, 2021.

[21] S. Tomasin and J. G. L. Hidalgo, "Virtual private mobile network with multiple gateways for b5g location privacy," in *2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)*, pp. 1–6, IEEE, 2021.

[22] P. Schmitt and B. Raghavan, "Pretty good phone privacy," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1737–1754, 2021.

[23] 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*, vol. 10, no. 7, pp. 5962–5976, 2023.

[24] V. O. Nyangaresi, A. J. Rodrigues, and S. O. Abeka, "Machine learning protocol for secure 5g handovers," *International Journal of Wireless Information Networks*, vol. 29, no. 1, pp. 14–35, 2022.

[25] J. Huang and Y. Qian, "A secure and efficient handover authentication and key management protocol for 5g networks," *Journal of Communications and Information Networks*, vol. 5, no. 1, pp. 40–49, 2020.

[26] 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*, vol. 18, no. 2, pp. 858–874, 2019.

[27] 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*, vol. 18, no. 3, pp. 1182–1195, 2019.

[28] 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*, vol. 122, no. 4, pp. 2925–2962, 2022.

[29] X. Yang, X. Huang, and J. K. Liu, "Efficient handover authentication with user anonymity and untraceability for mobile cloud computing," *Future Generation Computer Systems*, vol. 62, pp. 190–195, 2016.

[30] 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*, vol. 9, no. 3, pp. 1346–1358, 2022.
[31] 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*, pp. 1–8, Electronic Navigation Research Institute, 2022.
[32] 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.
[33] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," tech. rep., 1993.
[34] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 1, pp. 18–36, 1990.
[35] D. M. Nessett, "A critique of the burrows, abadi and needham logic," *ACM SIGOPS Operating Systems Review*, vol. 24, no. 2, pp. 35–38, 1990.
[36] 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*, pp. 147–166, Springer, 1996.
[37] C. Boyd and W. Mao, "On a limitation of ban logic," in *Workshop on the Theory and Application of of Cryptographic Techniques*, pp. 240–247, Springer, 1993.
[38] P. Teppo and K. Norman, "Security in 5g ran and core deployment," tech. rep., Apr. 2024.
[39] I. Ahmad, J. Pinola, I. Harjula, J. Suomalainen, E. Harjula, J. Huusko, and T. Kumar, "An overview of the security landscape of virtual mobile networks," *IEEE Access*, vol. 9, pp. 169014–169030, 2021.
[40] A. Sahai and B. Waters, "How to use indistinguishability obfuscation: Deniable encryption," tech. rep., and more. Cryptology ePrint Archive, Report 2013/454, 2013. http://eprint ..., 2013.
[41] 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*, pp. 651–681, Springer, 2022.
[42] N. Aviram, K. Gellert, and T. Jager, "Session resumption protocols and efficient forward security for tls 1.3 0-rtt," *Journal of Cryptology*, vol. 34, no. 3, p. 20, 2021.

APPENDIX A
MAPPING 5G-HO TO SECURE HO FRAMEWORK

In this section we demonstrate the universality and practical applicability of our formalised HO framework by mapping the existing 5G HO protocol into our syntax presented in Section IV. We further argue that in its current iteration, the 5G HO protocol lacks strong security guarantees against various attack vectors, as demonstrated by the lack of *forward secrecy* and *post compromise security* and other vulnerabilities as outlined by [1], [4]–[6].

The 3GPP 5G security standard outlines two main handover protocols, Xn- and N2-based, depending on the available network interfaces [1]. Despite their differences, both protocols are treated as equal alternatives in the 5G specification without clear recommendations. We focus on the Xn-based handover, which features direct communication between source and target base stations (known as gNBs within 5G), reducing message transmission to the core network (CN). In this section, we present a simplified 5G handover protocol that integrates all CN functions into a single abstract entity represented by the source gNB SRAN. This abstraction is justifiable since the internal communication of the CN guarantees confidentiality, integrity, authenticity, and replay protection. Our abstraction avoids unnecessary overhead from modeling CN

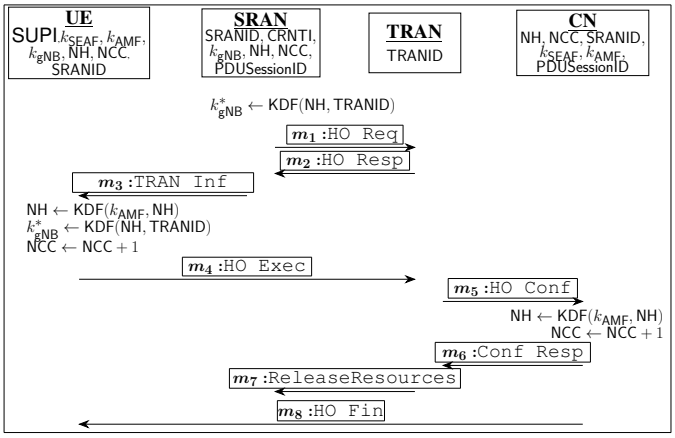

Fig. 7: 5G-HO protocol.

internal messages, which are assumed secure. Furthermore, the 3GPP architecture presumes a separation of Radio Access Network (RAN) and CN due to the precedent set by the legacy architectures of 3G and 4G. However, as [38], [39] point out co-locating RAN and CN architectures is a common deployment practice, particularly with the recent advances in implementation technology and the use-case specific requirements of IoT which necessitates low-latencies and high throughput. The current 3GPP standard fails to capture these deployment-specific nuances.

In 5G handover, four network entities are involved in executing the protocol, including user entity UE, source gNB (SRAN), target gNB (TRAN) and core network (CN). The execution of this protocol begins with prior keying parameters that were acquired during the 5G-AKA protocol. These parameters are: *i)* k_{SEAF}, a key shared between the UE and the CN; *ii)* k_{AMF}, a key derived from k_{SEAF}; *iii)* k_{gNB}, a session key generated by the UE and SRAN; *iv)* NH, an intermediate key along with its corresponding counter, NCC, which are derived by the UE and SRAN. The 5G handover protocol, illustrated in Figure 7, operates as follows [2]:

- $\boldsymbol{m_1}$: The SRAN requests a transfer of a UE to a TRAN by sending a newly derived session key (k^*_{gNB}) along with [TRANID, NCC,CRNTI, PDUSessionID]. We map m_1 to the Supp phase of our framework.
- $\boldsymbol{m_2}$: Upon acceptance of the UE by the TRAN, a handover acknowledgement response is issued. We map m_2 to the Supp phase of our framework.
- $\boldsymbol{m_3}$: After receiving m_2, the TRAN encrypts it along with their identity (TRANID) using the session key (k_{gNB}) and then forwards it to the UE. We map m_3 to the Prep phase of our framework.
- $\boldsymbol{m_4}$: Upon receipt of the message, the UE derive a new session key (k^*_{gNB}) to encrypt the ReleaseResources message and send it to the TRAN. We map m_4 to the Cont phase of our framework.
- $\boldsymbol{m_5 \& m_6}$: The TRAN and CN have established a mutual agreement on session identifiers and temporary keys. We map m_5 and m_6 to the Supp phase of our framework.

[2]Details of the full protocol can be found in [TS23.502] and [TS 38.300].

– m_7: The TRAN informs the SRAN that all resources allocated to the UE can now be released. We map m_7 to the Supp phase of our framework.
– m_8: Finally, the CN informs the UE of a successful handover and registration. We map m_8 to the Prep phase of our framework.

In Figure 8 we present the mapping of the conventional 5G Handover protocol [1] to our handover framework. In our mapping we consider all communications with CN as internal operations of source SRAN and we treat SRAN as an honest party who voluntarily deletes all state information and related keys pertaining to the now completed handover. Consequently, we treat the transactions taking place between the target TRAN and CN as an execution of the Supp algorithm. Interestingly, the execution of the Supp, Prep and Cont algorithms overlap and occur simultaneously, which ensures that our mapping preserves the correctness of the original protocol flow.

APPENDIX B
MAPPED HO CONSTRUCTIONS

In Figure 9 we have illustrated known HO constructions within our formalised framework.

APPENDIX C
FULL KIND PROOFS FOR StrongHO

Here we provide detailed full proof for KIND security of our StrongHO construction.

Proof. We split the analysis into three cases:
– **Case 1:** Test session does not UT-match another session and $\pi_i^s.\rho = \mathtt{U}$
– **Case 2:** Test session does not UT-match another session and $\pi_i^s.\rho = \mathtt{T}$
– **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 $\pi_i^s.\rho = \mathtt{U}$. By the definition of the case (and the cleanness predicate defined in Definition 1), 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_j^t such that $ctxt_{\mathtt{U}} \in \pi_i^s.T_{\mathtt{H}}$, $ctxt'_{\mathtt{U}} \in \pi_j^t.T_{\mathtt{H}}$, $ctxt_{\mathtt{U}} = ctxt'_{\mathtt{U}}$, and $epk_{\mathtt{U}} \in \pi_i^s.T_{\mathtt{H}}$, $epk'_{\mathtt{U}} \in \pi_j^t.T_{\mathtt{H}}$, $epk_{\mathtt{U}} = epk'_{\mathtt{U}}$.

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

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 predicate in Definition 1, $\mathcal{A}$ cannot issue Compromise queries before $\pi_i^s.\alpha = \mathtt{accept}$, and since bk is already uniformly random and independent, this change is sound. 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 initialises an auth challenger $\mathcal{C}_{\mathsf{auth}}$. Whenever $\mathcal{C}$ is required to encrypt/decrypt using $\widetilde{tk}$, $\mathcal{B}_2$ instead queries auth to $\mathcal{C}_{\mathsf{auth}}$. By **Game 4** $\widetilde{tk}$ is uniformly random and independent, 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 13. We note that the ciphertext c_1 contains (and thus authenticates) the signature σ_0, which itself is computed over all messages received by S from the U. Thus, U now aborts if they complete the preparation phase without a 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 $\mathtt{event}_{\mathtt{T}}$ that occurs if the Test session π_i^s sets $\alpha = \mathtt{accept}$ without an honest UT-match. Thus: $\mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) \leq \Pr(\mathtt{event}_{\mathtt{T}}) + \mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda)$. We note that by definition of the case, π_i^s never has a UT-match and thus $\mathsf{Adv}^{\mathcal{A}}_{G_7}(\lambda) = 0$, since $\mathcal{A}$ can never test a session that aborts before $\alpha \leftarrow \mathtt{accept}$. In what follows, we bound $\Pr(\mathtt{event}_{\mathtt{T}})$.

Game 8 In this game, $\mathcal{C}$ replaces ck in $ctxt_{\mathtt{T}}$ 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 ikcca PKE challenger.

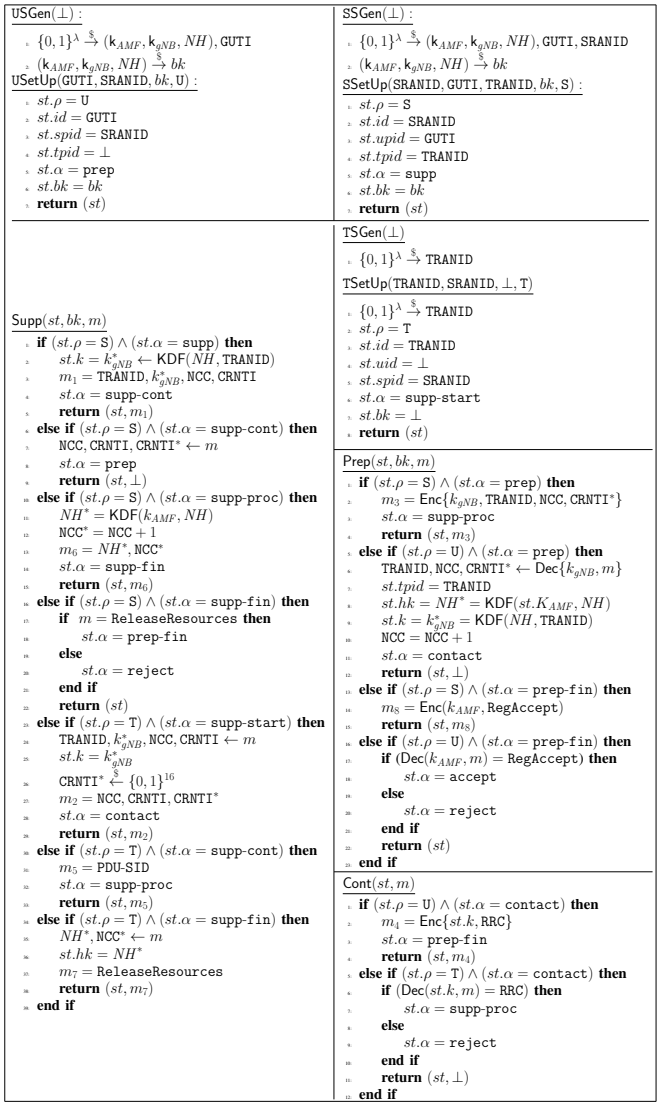

Fig. 8: The 5G Handover protocol as a Secure Handover Scheme.

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

Game 9 Similar to **Game 4**, in this game $\mathcal{C}$ replaces the derived keys $k, hk, mk' = \mathsf{KDF}(k_{\mathtt{U}}, \widetilde{ck})$ 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. 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, by defining a reduction $\mathcal{B}_5$ that interacts with a MAC challenger $\mathcal{C}_{\mathsf{eufcma}}$. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_9}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_{10}}(\lambda) + \mathsf{Adv}^{\mathcal{B}_5,\mathsf{eufcma}}_{\mathsf{MAC}}(\lambda)$. 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 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 $\mathsf{Adv}^{\mathsf{KIND},\mathsf{clean}_{\mathsf{str\text{-}kind}},\mathcal{A}}_{\mathsf{StrongHO},n_P,n_S,C_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$

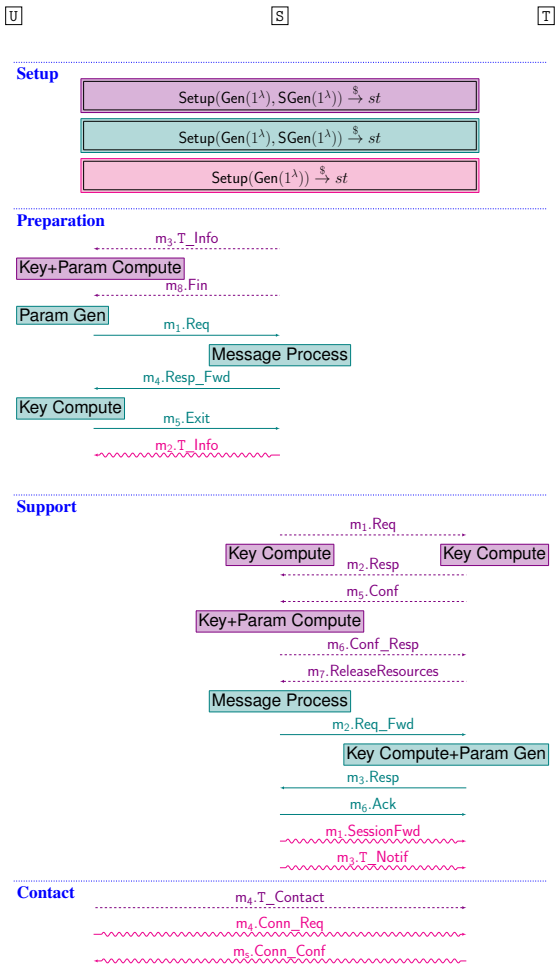

Fig. 9: Mapping existing HO schemes to our framework.
5G-HO | LDACS-HO | CPDLC-HO

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 $\mathcal{B}_6$ initialises 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. 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 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 $\mathsf{PPRF}.\mathcal{C}(epk_{\mathtt{U}} \| ctxt_{\mathtt{T}})$, 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 11. 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 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 $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) + \mathsf{Adv}^{\mathcal{B}_7,\mathsf{eufcma}}_{\mathsf{MAC}}(\lambda)$.

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 1, $\mathcal{A}$ 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 $\mathcal{C}_{\mathsf{eufcma}}$ as in **Case 1**, **Game 10**. Thus $\mathcal{A}$ cannot modify messages to π_u^k and we have $\mathsf{Adv}^{\mathcal{A}}_{G_5}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_6}(\lambda) + \mathsf{Adv}^{\mathcal{B}_{10},\mathsf{eufcma}}_{\mathsf{MAC}}(\lambda)$. 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 ist 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-cpa 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-cpa KEM challenger, and the computation of $k_{\mathtt{U}}$ with the output key. Detecting the replacement of $k_{\mathtt{U}}$ implies an efficient

distinguishing PPT algorithm $\mathcal{A}$ against ind-cpa security of KEM. Thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + \mathsf{Adv}^{\mathcal{B}_{11},\mathsf{ind\text{-}cpa}}_{\mathsf{KEM}}(\lambda)$.

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 $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) + \mathsf{Adv}^{\mathcal{B}_{12},\mathsf{prf}}_{\mathsf{PRF}}()$.

Here we emphasise 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 $\mathsf{Adv}^{\mathcal{A}}_{G_3}(\lambda) = 0$.

Appendix D
Security Definition for Source Privacy

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

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

1: $b \xleftarrow{\$} \{0,1\}$
2: **for** $i = 1$ **to** n_P **do**
3: $pk_i, sk_i \leftarrow \mathsf{Gen}$
4: $\mathbf{ASK}_i \leftarrow \mathtt{false}$
5: $ctr_i \leftarrow 1$
6: **for** $j = 1$ **to** n_S **do**
7: $\mathbf{SSK}^j_i \leftarrow \mathtt{false}$
8: $\mathbf{SK}^j_i \leftarrow \mathtt{false}$
9: **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: **return** $b \xleftarrow{\$} \{0,1\}$
14: **else**
15: **return** $(b' = b)$
16: **end if**

$\mathsf{SendTest}(m)$

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

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

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

Fig. 10: The source privacy security experiment for secure handover 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. $\mathbb{Q}$ denotes the set of all queries used in the experiment, i.e. $\mathbb{Q} = \{\mathsf{Send}, \mathsf{Corrupt}, \mathsf{Compromise}, \mathsf{Reveal}, \mathsf{Create}, \mathsf{TestSource}, \mathsf{SendTest}\}$.

Definition 8 (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.

Appendix E
SPRIV Cleanness Predicate for Strong Handover Protocol

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.

Definition 9 (Strong Handover SPRIV cleanness predicate)**.** *A session* π^s_i *such that* $\pi^s_i.\alpha = \mathtt{accept}$ *in the security experiment defined in Figure 10 is* $\mathsf{clean}_{\mathsf{str\text{-}spriv}}$ *if all of the following conditions hold:*

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

Appendix F
Cryptographic Assumptions

In this section we define the cryptographic formalism and assumptions that we use to build our secure handover schemes in Section V.

Definition 10 (prf Security)**.** *A pseudo-random function family is a collection of deterministic functions* $\mathsf{PRF} = \{\mathsf{PRF}_\lambda : \mathcal{K} \times \mathcal{I} \rightarrow \mathcal{O} : \lambda \in \mathbb{N}\}$*, 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 secure if for all PPT algorithms $\mathcal{A}$,* $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda)$ *is negligible in the security parameter λ.*

A puncturable pseudo-random function is a special instance of a pseudo-random function (PRF), that facilitates the computation of punctured keys, which prohibits evaluation on inputs that have been punctured. We refer to the definition of puncturable pseudo-random functions and its security from [40], but restrict our attention to PPRFs with deterministic puncturing algorithms as defined by [41].

Definition 11 (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 $sk \in \mathcal{K}$.*
- $\mathsf{Eval}(sk, x) \rightarrow y/\perp : \mathsf{Eval}$ *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, $\perp$.*
- $\mathsf{Punc}(sk, x) \rightarrow sk' : \mathsf{Punc}$ *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 Strong HO 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. [42] formally defines *invariant puncturing* as follows:

Definition 12 (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 [42], which we have presented in Figure 11 .

Definition 13 (ae-auth Security)**.** *An AE scheme* AE *is a triple of algorithms* $\mathsf{AE} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *with an associated keyspace $\mathcal{K}$ 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* $\mathsf{AE.Enc}(k, M)$ *the AE encryption algorithm that takes as input a key $k \in \mathcal{K}$ and a message $M \in \mathcal{M}$ and outputs a ciphertext $C \in \{0,1\}^*$. We denote by* $\mathsf{AE.Dec}(k, C)$ *the AE decryption algorithm that takes as input a key $k \in \mathcal{K}$ 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* $\mathsf{AE.Dec}(k, \mathsf{AE.Enc}(k, M)) = M$ *for all k, M in the appropriate space.*

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

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

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

Fig. 11: Adaptive-random PPRF security experiment.

Definition 14 (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 $\forall (pk, sk)$ 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 the* ind-cpa *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) 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* 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.*

Now we strengthen our assumptions by defining ind-cca security for KEMa:

Definition 15 (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 $\mathcal{K}$, 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* KEM *is post-quantum* 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* KEM *is classically* 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.*

Next, we turn to defining eufcma security for message authentication codes (MACs).

Definition 16 (Message Authentication Code (MAC) security)**.** *A message authentication code (*MAC*) scheme is a tuple of algorithms* $\mathsf{MAC} = \{\mathsf{KGen}, \mathsf{Tag}\}$ *where:*

- KGen *is a probabilistic key generation algorithm taking input a security parameter λ and returning a symmetric key k.*
- Tag *is a deterministic 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.*

Security is formulated via the following game that is played between a challenger $\mathcal{C}$ and an algorithm $\mathcal{A}$:

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 unforgeability property of a MAC* MAC *under chosen-message attack to be:*

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

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

Finally, we turn to defining classical eufcma security for digital signatures.

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

- KGen *is a probabilistic key generation algorithm taking input a security parameter λ and returning a public key pk and a secret key sk.*
- Sign *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.*
- Vfy *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* $\mathsf{SIG.Vfy}(pk, m, \mathsf{SIG.Sign}(sk, m)) = 1$*. Security is formulated via the following game that is played between a challenger $\mathcal{C}$ and an algorithm $\mathcal{A}$:*

1) The challenger samples $pk, sk \xleftarrow{\$} \mathcal{K}$
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:*

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

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

Definition 18 (Key Indistinguishability of Public Key Encryption)**.** *A public key encryption (*PKE*) scheme is a tuple of algorithms* $\mathsf{PKE} = \{\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}\}$ *where:*

- KGen *is a probabilistic key generation algorithm taking input a security parameter λ and returning a public key pk and a secret key sk.*
- Enc *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.*
- Dec *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* $(pk, sk) \xleftarrow{\$}$ PKE.KGen*, we have* $\mathsf{PKE.Dec}(sk, \mathsf{PKE.Enc}(pk, m)) = m$*. Indistinguishability of keys under chosen ciphertext attack (*ikcca*) Security is formulated via the following game that is played between a challenger* $\mathcal{C}$ *and an algorithm* $\mathcal{A}$*:*

1) *The challenger samples* $(pk_0, sk_0), (pk_1, sk_1) \xleftarrow{\$} \mathcal{K}(\lambda)$ *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:*

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

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*.*