PGUP: Pretty Good User Privacy for 5G-enabled Secure Mobile Communication Protocols

Rabiah Alnashwan*
Imam Mohammad Ibn Saud Islamic University
University of Sheffield
ralnashwan@imamu.edu.sa

Prosanta Gope
University of Sheffield
p.gope@sheffield.ac.uk

Benjamin Dowling
King's College London
benjamin.dowling@kcl.ac.uk

Yang Yang*†
National University of Singapore
y.yang@u.nus.edu

Abstract

With the proliferation of 5G networks, it is essential to prioritise robust security and seamless compatibility with existing infrastructure. The Authentication and Key Agreement (AKA) and Handover (HO) protocols are crucial in securing communication links and maintaining user privacy in 5G networks. While 5G-AKA represents a significant improvement over its predecessors, it still cannot achieve some important security features, such as perfect forward security (PFS) and forward privacy (PFP), leaving data confidentiality and user privacy susceptible to compromise. Moreover, linkability vulnerabilities in the 5G-AKA pose additional privacy concerns, particularly in the face of active adversaries seeking to compromise user anonymity. To enhance the security and privacy of 5G protocols (5G-AKA and 5G-HO) , we aim to achieve PFS and PFP while aligning with 5G's symmetric-key foundations. In this article, we introduce Pretty Good User Privacy (PGUP), a *novel* symmetric-based scheme aimed at addressing security and privacy vulnerabilities in the current 5G-AKA and HO protocols. In this article, we introduce a *new variant of Puncturable Key Wrapping* (i.e., PKW$^+$), which allows us to ensure PFS and PFP while maintaining resilience against DoS (desynchronization) attacks in our proposed protocols. We demonstrate that our proposed scheme is resilient against all the essential security threats by performing a comprehensive formal security analysis. We also conduct relevant experiments to show the efficiency of the proposed scheme.

Keywords

PGUP, 5G, Authentication and key agreement, Handover, User privacy, Unlinkability, Perfect forward security

1 Introduction

The advent of 5G technology marks a significant milestone in wireless communication, offering unprecedented speed, capacity, and connectivity. This transformative shift necessitates a rigorous examination of the associated security and privacy frameworks. Central to 5G security are two key protocols: the Authentication and Key Agreement (AKA) and the Handover (HO) protocols [1]. The 5G-AKA protocol is primarily responsible for authenticating users and granting access to network services, ensuring data confidentiality and integrity in standard network operations. Complementing this, the 5G-HO protocol serves a similar function but is designed explicitly for roaming users. Together, these protocols offer a more robust security infrastructure compared to previous mobile communication generations like 3G and 4G. However, while 5G-AKA represents a significant improvement over its predecessors, it still falls short in achieving some critical security features [5, 11, 12]. One of the most significant vulnerabilities is that the current 5G-AKA protocol doesn't support Perfect Forward Secrecy (PFS) and Perfect Forward Privacy (PFP). The absence of PFS and PFP in current 5G protocols leaves users vulnerable to different types of attacks if long-term keys are compromised. PFS ensures that even if long-term secrets are compromised, past session keys remain secure, protecting previous communications. PFP, on the other hand, is crucial for maintaining user anonymity over time. Without it, a compromise of long-term keys could allow an adversary to reveal users' identities. Additionally, active adversaries pose a significant linkability vulnerability. This specific privacy concern grants the adversary the ability to link a user's activities throughout their connection to the network, unveiling their digital footprint and may lead to compromising user anonymity. In contrast to passive adversaries, active adversaries take a proactive stance, engaging with the network by attempting to actively modify or manipulate data in transit, aiming to compromise security or disrupt normal operations. Achieving privacy and security (PFP, unlinkability, PFS) can be relatively straightforward in an asymmetric setting; however, asymmetric solutions often involve complex cryptographic operations that are computationally intensive. This complexity poses a significant challenge in the context of 5G networks, where User Equipment (UE) often operates under resource constraints. Many UE devices, such as IoT sensors, wearables, and low-power mobile devices, have limited computational capabilities, restricted memory, and constrained energy resources. These limitations make the implementation of complex asymmetric cryptographic operations unpractical. Given these constraints, accomplishing PFS, PFP, and unlinkability within a symmetric setting presents unique challenges, but offers a more viable path for resource-limited UE devices. In symmetric systems, addressing these features requires frequent key updates that introduce significant overhead and potential synchronization problems.

*Corresponding Author.
†Contributed to the implementation of the PGUP system using OpenAirInterface.

Proceedings on Privacy Enhancing Technologies 2025(2), 450–478

https://doi.org/10.56553/popets-2025-0071

However, given the symmetric nature of 5G operations and the efficiency of symmetric cryptography on resource-constrained devices, it is crucial to explore symmetric-based solutions that require minimal infrastructural changes while providing robust security and privacy guarantees. Significant research efforts have been dedicated to addressing security and privacy concerns within 5G networks, ranging from enhancements to existing 5G-AKA and HO protocols to the proposition of entirely new protocols. However, our comprehensive literature review reveals a critical gap: no current solution simultaneously addresses all the essential security and privacy challenges in 5G networks, particularly in a symmetric-based setting, as discussed in Section 2. Therefore, there is a need for a solution that provides PFS, PFP, unlinkability, and secure revocation while supporting universal handover, all within the constraints of symmetric cryptography and the computational limitations of UE devices. This complex set of requirements, coupled with the necessity to maintain compatibility with existing infrastructure and respect the limited computational capabilities of UE, motivates our research. To bridge this gap, we utilize a *symmetric-based* primitive called Puncturable Key Wrapping (PKW) [4] and develop a *new* variant tailored specifically for the 5G environment. Building upon this, we introduce Pretty Good User Privacy (PGUP), an innovative *symmetric-based* scheme designed to address security and privacy vulnerabilities in the current 5G-AKA and 5G-HO protocols. Our key contributions include:

- The *first* standalone symmetric-based solution achieves PFS, PFP, user unlinkability, secure revocation, and seamless universal handover;
- Design a *new variant* of puncturable key wrapping (i.e., PKW^+), tailored specifically for 5G, to achieve PFS, PFP and unlinkability in a symmetric base setting;
- The proposed solution aims to maintain conceptual alignment with 5G protocol structures, potentially facilitating integration with minimal modifications to existing infrastructure;
- A comprehensive formal security analysis of our proposed scheme;
- A comparative performance evaluation of PGUP with the conventional 5G-AKA and HO protocols demonstrating the cost-effectiveness of the proposed PGUP scheme.

2 Related Works and Motivation

Extensive research on 5G authentication and handover protocols, particularly 5G-AKA, has revealed significant security and privacy vulnerabilities. Peltonen et al. [20], Basin et al. [5], and Cremers and Dehnel-Wild [12] have conducted comprehensive formal security analyses of 5G-HO protocol. These studies have uncovered under-specified security requirements and various vulnerabilities in current 5G-AKA and 5G-HO protocols [1]. Key findings include traceability attacks against 5G-AKA in the presence of active adversaries [5], confusion attacks exploiting identity misbinding for impersonation [12], and logical vulnerabilities compromising sequence number confidentiality due to XOR usage and lack of randomness [9]. Additionally, Braeken [11] identified that a simple identity replay attack, previously demonstrated against various AKA protocols [14], also poses a threat to 5G-AKA. In response to these challenges, Schmitt and Raghavan [24] introduced PGPP, a cellular architecture enhancement aimed at enhancing user identity and location privacy. By separating network connectivity from authentication and billing processes, PGPP enables carriers to offer services without needing to know the identity or location of their users while still ensuring their authorization for network access. However, to achieve their goals, the authors proposed some infrastructural-level changes (e.g., the inclusion of new entities) that require major changes in the current 5G infrastructure. Besides, the proposed scheme does not address specific protocol-level security issues. Given these architectural limitations, other researchers have also focused on more targeted solutions that can be implemented within the existing 5G framework to enhance the 5G-AKA and 5G-HO protocols. These proposed enhancements can be broadly categorised into symmetric-based and asymmetric-based approaches. Table 1 provides a comparative analysis of prior research and our scheme, focusing on each solution's security and privacy features.

Table 1: Features comparison with State-of-the-art Protocols.

Type	Scheme	MA	PFS	PFP	Unlink	SRM	UHO
Symmetric-key-based	5G [1]	✓	✗	✗	✗	✗	✗
	AKA'[26]	✓	✗	✗	✓	✗	✗
	ReHand[13]	✓	✗	✗	✓	✓	‡
Asymmetric(Public)-key-based	*AAKA* [27]	✓	✗	✗	✓	✗	✗
	AKA^+ [16]	✓	✗	✗	✓	✗	✗
	Protocol of [2]	✓	✓	✗	✓	✓	‡
Ours (Symmetric-key-based)	PGUP	✓	✓	✓	✓	✓	✓

MA:Mutual Authentication, **PFS**: Perfect Forward Secrecy, **PFP**: Perfect Forward Privacy, **Unlink**: Unlinkability, **SRM**: Secure Revocation Management, **UHO**: Universal HO, ‡: Region-based HO

Symmetric-key-based solutions. Several symmetric-based solutions have been proposed to enhance the 5G-AKA protocol and address privacy concerns. The standard 5G protocol [1], while providing Mutual Authentication (MA), lacks PFS along with other advanced security features. AKA' [26] improves upon the standard by addressing the linkability issue through minor modifications to the current 5G-AKA protocol, achieving high compatibility and unlinkability. However, as evident from Table 1, AKA' does not achieve PFS, leaving a critical security gap. Similarly, ReHand [13], which introduces a region-based HO protocol designed to protect roaming users' anonymity and ensure unlinkability, also falls short in providing PFS. ReHand offers additional features like fast revocation management and seamless handovers within specific regions (‡in Table 1), but its practical applicability is limited due to the lack of support for region-based handovers in the current 5G infrastructure. Achieving PFS in symmetric-key cryptosystems presents unique challenges compared to asymmetric settings. Existing solutions predominantly rely on ephemeral Diffie-Hellman key exchange, an asymmetric primitive. While effective in providing PFS, this approach can introduce additional computational complexity compared to purely symmetric-key operations. As shown in Table 1, previous symmetric-based solutions have struggled to achieve PFS in the context of 5G networks. This gap in the field has motivated numerous research efforts to focus on asymmetric-based solutions to meet this security requirement.

Asymmetric (Public)-key-based solutions. Recent research has proposed innovative asymmetric-based solutions to address the critical security and privacy challenges in 5G networks. Alnashwan

et al. [2] introduced a region-based HO with AKA protocols, which offer several key features: secure user revocation, PFS, and unlinkability within region-based HO. However, this protocol is limited to region-based HO and does not support universal handover across the entire network. Additionally, research has delved into tracking users' digital footprint within the cellular network. Yu et al. [27] introduced AAKA, an AKA protocol leveraging cryptographic assumptions such as Decisional Diffie-Hellman, zero-knowledge proof, BBS signatures, Keyed-Verification Anonymous Credential, and ElGamal Encryption. Similarly, The AKA$^+$ solution improves the privacy of 5G-AKA while satisfying its design and efficiency constraints, effectively dealing with the IMSI catcher's attacks and achieving a certain degree of unlinkability. However, it's crucial to note that while AAKA and AKA$^+$ aim for privacy, their scope is limited to the AKA protocol, leaving privacy concerns unaddressed during handovers.

Despite the advancements in 5G security and privacy protocols, including both symmetric and asymmetric solutions, significant challenges remain. The integration of various cryptographic primitives in protocols like AAKA [27], AKA$^+$ [16], and others [2] has pushed the boundaries of what's possible in securing 5G networks. These solutions have made strides in addressing issues such as user anonymity, unlinkability, and protection against specific attacks like IMSI catchers. However, it is crucial to note that many of these protocols rely heavily on asymmetric cryptographic elements, which is not ideal for the predominantly symmetric-based 5G infrastructure. This mismatch is particularly problematic for resource-constrained devices commonly found in Internet of Things (IoT) ecosystems, such as sensors, wearables, and low-power embedded systems. These devices often have limited computational power, memory, and energy resources, making complex asymmetric cryptographic operations computationally expensive and energy-intensive. Furthermore, none of the existing protocols in the literature have fully achieved *PFP*, and many have not attained *PFS* either. Both of these features are crucial for guaranteeing users' long-term security and privacy, even in scenarios where encryption keys might be compromised in the future. This gap in current solutions, coupled with the need for solutions that align with the symmetric nature of 5G systems, continues to drive research efforts in the field.

3 5G Authentication (5G-AKA)

Mobile network authentication ensures that only legitimate users can access network services while protecting their privacy and securing their communications. This process involves three key parties: the UE such as mobile phones or IoT devices seeking network access; the base station (gNB) that provides radio coverage and relays authentication messages; and the Core Network (CN) that verifies user credentials and manages authentication. When a user wants to connect to the network, two fundamental challenges must be addressed: the network needs to verify the user is legitimate and has the right to access services, while the user needs to verify they are connecting to a genuine network operator. Additionally, both parties need to establish shared secret keys for securing subsequent communications. To accomplish these goals, the 3rd Generation Partnership Project (3GPP) outlines two primary AKA protocols: 5G-AKA [1] and Extensible Authentication Protocol (EAP-AKA') [3]. EAP-AKA' serves as a mechanism for authentication and session key distribution. Both EAP-AKA' and 5G-AKA utilize the latest version of the 3GPP AKA for secure authentication and key agreement. In essence, both protocols rely on the same underlying infrastructure and specifications of 3GPP-AKA, employing identical cryptographic protocol constructions. Hence, this work focuses primarily on the 5G-AKA to maintain consistency. The 5G-AKA employs a collection of seven distinct symmetric-key algorithms labelled as f_1 through f_5, as well as f_1^* and f_5^*. Specifically, f_1, f_2, and f_1^* serve as message authentication functions, while f_3, f_4, f_5, and f_5^* are employed as key derivation functions. Furthermore, 5G also employs SHA256 for hashing responses between the UE and the CN. In 5G-AKA, there is a *challenge-response* phase and an optional *resynchronisation* phase, which is performed if the SQN gets out of sync between UE and CN. The 5G-AKA protocol, illustrated in the Appendices-Figure 8, operates through the following steps: **Step 1**: This phase starts when a UE sends an authentication request using SUCI (Subscription Concealed Identifier, an encrypted version of SUPI-the user's permanent identity) or 5G-GUTI to CN. **Step 2**: Upon receiving the authentication request, the CN de-conceals the SUCI and retrieves the SUPI. Afterwards, CN computes the expected response $xRES^*$ and the authentication challenge using a random nonce R, an authentication token ($AUTN$), challenge expected response ($HXRES$) and K_{SEAF}, and sends it to the gNB. The $AUTN$ verifies the challenge's freshness, combining the Message Authentication Code (MAC), the nonce R, with the corresponding sequence number SQN of UE. The $HxRES^*$ is only sent to the gNB, which is the hashed value of the actual RES^*. **Step 3**: Upon receiving the authentication challenge, the gNB stores $HxRES^*$, as well as the K_{SEAF}, then forwards the challenge to the UE. **Step 4**: Next, the UE verifies the authenticity and freshness of the challenge by (1) Extracting the xSQN (the incremented value of SQN) and MAC from AUTN.(2) Verify the correctness of MAC and respond with "Mac-failure" if errors occur. (2) Verify the freshness of SQN, i.e. ensuring SQN < xSQN. If all verifications hold, the UE computes a challenge-response RES^* and derives K_{SEAF} to secure the channel used with the gNB, then sends RES^* to gNB. **Step 5**: Upon receiving the challenge-response RES^*, the gNB computes the hash value of RES^* ($HRES^*$) and compares it with $HxRES^*$ received from the CN. If $HRES^*$ equals $HxRES^*$, then the authentication is successful. Subsequently, the gNB forwards RES to the CN containing SUCI or SUPI and the gNB name. **Step 6**: Upon receiving the RES^* from the gNB, the CN checks if RES equals $xRES^*$, hence confirming the success of the authentication procedure, then notifying the gNB of the decision.

4 Cryptographic Primitives

PGUP has been designed to maintain conceptual alignment with 5G protocol structures, potentially facilitating integration with minimal modifications to existing infrastructure. Therefore, we have adopted the same set of asymmetric and symmetric-key algorithms to achieve this alignment. These include the Elliptic Curve Integrated Encryption Scheme (ECIES) and $f_1, \ldots, f5, f_1^*, f_5^*$. Additionally, we have developed a new variant of Puncturable Key Wrapping (PKW$^+$) that is specifically tailored to 5G infrastructure,

which utilizes symmetric-based primitives to enhance security measures. In this section, we will explain the significance of PKW$^+$ in building PGUP, as well as the reasoning behind the development of this new variant of the original PKW.

4.1 Puncturable Key Wrapping

The puncturable key-wrapping (PKW) scheme introduced in [4] provides a symmetric-key hierarchy with an updatable key, ensuring PFS in a symmetric-key setting. PKW achieves this by combining symmetric-key wrapping with a puncturing algorithm, allowing the wrapping key to be updated while rendering the wrapped key irrecoverable.

Definition 1 (Puncturable Key Wrapping). *The PKW consists of a tuple of four algorithms PKW:*{KGen, Wrap, Unwrap, Punc} *associated with four sets: the secret-key space $\mathcal{SK}$, the tag space $\mathcal{T}$, the header space $\mathcal{H}$ and the wrap-key space $\mathcal{K}$.*

- **KGen()** $\rightarrow$ sk*: a probabilistic algorithm that takes no inputs and outputs a secret key sk* $\in \mathcal{SK}$.
- **Wrap**$(sk, T, H, K) \rightarrow C/\perp$*: a deterministic wrapping algorithm takes an input of a secret key sk* $\in \mathcal{SK}$ *, a tag* $T \in \mathcal{T}$*, a header* $H \in \mathcal{H}$*, and a key* $K \in \mathcal{K}$ *and outputs either a ciphertext* $C \in \{0,1\}^*$ *or* $\perp$ *for failure.*
- **Unwrap**$(sk, T, H, C) \rightarrow K/\perp$*: a deterministic unwrapping algorithm takes an input of a secret key sk* $\in \mathcal{SK}$ *, a tag* $T \in \mathcal{T}$*, a header* $H \in \mathcal{H}$*, and a ciphertext* $C \in \{0,1\}^*$ *and outputs either a key* $K \in \mathcal{K}$ *or* $\perp$ *for failure.*
- **Punc**$(sk, T) \rightarrow sk'$*: a deterministic puncturing algorithm takes an input of a secret key sk* $\in \mathcal{SK}$ *and a tag* $T \in \mathcal{T}$ *and returns an updated secret key sk'* $\in \mathcal{SK}$.

The PKW primitive combines key wrapping with puncturing to achieve forward security. The wrapping mechanism uses three key components: a tag (T) that uniquely identifies each wrapped key, a header (H) that can authenticate associated metadata, and the actual key (K) to be wrapped. The critical security property comes from puncturing: when a key is punctured on a tag T; it can no longer unwrap any key that used that tag – even if the wrapping happened before puncturing. This irreversible property enables applications like secure file deletion and forward-secure session tickets. As described by Backendal et al.[4], PKW achieves puncturing by building on a puncturable PRF (PPRF) and an AEAD scheme, defined in Appendices-A.4. The master key is the PPRF key, and wrapping a key K with tag T works by first deriving a one-time AEAD key via $\mathsf{sk}_a \leftarrow \mathsf{PPRF.Eval}(\mathsf{sk}_\mathsf{p}, \mathsf{T})$, then using this to encrypt K. The puncturing operation PKW.Punc is implemented by directly calling the underlying PPRF puncturing operation: $\mathsf{PKW.Punc}(\mathsf{sk}_\mathsf{p}, \mathsf{T}) := \mathsf{PPRF.Punc}(\mathsf{sk}_\mathsf{p}, \mathsf{T})$. This makes the AEAD key for tag T unrecoverable, effectively invalidating any wrapped key using that tag.

While the PKW scheme guarantees PFS in symmetric-key settings, its immediate application to 5G-enabled mobile communication could face challenges due to potential security reductions linked to *key-reuse*, as discussed in [19]. Cryptographic algorithms are crafted with specific security properties and assumptions, and using a key designed for one algorithm in a different context could introduce vulnerabilities, jeopardizing the overall security of the system. To illustrate this, consider the following use case in 5G-AKA (illustrated in Appendices-Figure 8): If we were to use PKW *directly* in the 5G-AKA protocol, the same key K would be used for:

(1) In **PKW** for wrapping/encrypting the user's identity (SUPI);
(2) In **KDF** for deriving keys (AK, HK, K_{SEAF});
(3) In **MAC and Hash** for generating MAC and the expected response ($xRES^*$), respectively.

This multi-purpose use of K, employed across various cryptographic schemes such as PKW, KDF, MAC, and Hash, could lead to potential vulnerabilities. This interdependence of security properties, stemming from the use of a single key, could significantly impact the overall robustness of the protocol. The PKW$^+$ scheme addresses these concerns by introducing a separate key derivation function. This modification effectively separates the key used for wrapping from the keys used in other parts of the protocol:

- The long-term key K is used only for PKW$^+$ scheme (Wrap, Unwrap, Drv and Punc).
- Separate derived keys are used for other cryptographic operations in the protocol.

This separation ensures mutual protection: compromise of the derived keys reveals no information about the long-term key K, and compromise of K reveals nothing about the derived keys. By introducing PKW$^+$, we maintain the PFS guarantees of the original PKW while mitigating the risks associated with *key-reuse*. This approach aligns with the principle of key separation in cryptographic protocol design, enhancing the overall security of the 5G-AKA protocol.

4.2 Proposed Puncturable Key Wrapping(PKW$^+$)

PKW$^+$ is a customised variant of PKW, specifically designed for the 5G environment. It retains the essential functionalities and PFS properties of PKW while incorporating key derivation capabilities. The main modifications include the integration of a Key Derivation Function (KDF) into the existing algorithms and the introduction of a new algorithm (Drv) for deriving keys for use in other contexts. These enhancements, illustrated in **Figure 1**, mitigate risks associated with key-reuse, thereby strengthening overall security*.

Definition 2 (Puncturable Key Wrapping$^+$). *The* PKW$^+$ *consists of a tuple of five algorithms, denoted as* PKW$^+$:{KGen, Wrap, Unwrap, Punc, Drv}, *associated with five sets: the secret-key space $\mathcal{SK}$, the tag space $\mathcal{T}$, the additional data space $\mathcal{AD}$, message space $\mathcal{M}$, and the derive-key space $\mathcal{K}$.*

- **KGen()** $\rightarrow$ sk*: a probabilistic algorithm that takes no inputs and outputs a secret key sk* $\in \mathcal{SK}$.
- **Wrap**$(sk, T, AD, m) \rightarrow C/\perp$*: a wrapping algorithm takes an input of a secret key sk* $\in \mathcal{SK}$ *, a tag* $T \in \mathcal{T}$*, an additional data* $AD \in \mathcal{AD}$*, and a message* $m \in \mathcal{M}$ *and outputs either a ciphertext* $C \in \{0,1\}^*$ *or* $\perp$ *for failure.*
- **Unwrap**$(sk, T, AD, C) \rightarrow m/\perp$*: a deterministic unwrapping algorithm takes an input of a secret key sk* $\in \mathcal{SK}$ *, a tag* $T \in \mathcal{T}$*, an additional data* $AD \in \mathcal{AD}$*, and a ciphertext* $C \in \{0,1\}^*$ *and outputs either a message* $m \in \mathcal{M}$ *or* $\perp$ *for failure.*

*The security proof of PKW$^+$ can be found in the Appendix A.

- **Punc$(sk, T) \rightarrow sk'$**: *a deterministic puncturing algorithm takes an input of a secret key* $sk \in \mathcal{SK}$ *and a tag* $T \in \mathcal{T}$ *and returns an updated secret key* $sk' \in \mathcal{SK}$.
- **Drv$(sk, T, AD) \rightarrow k$**: *a deterministic key derivation function algorithm takes an input of a secret key* $sk \in \mathcal{SK}$, *a tag* $T \in \mathcal{T}$ *and additional data* $AD \in \mathcal{AD}$ *and returns a derived secret key* $k \in \mathcal{K}$.

Correctness. The PKW$^+$ requires that the wrapped message can be successfully unwrapped from the wrapping ciphertext, even if the secret key has been punctured on various tags, except for the specific tag utilized in the wrapping of that particular message. Formally, we require that for all $T \in \mathcal{T}, AD \in \mathcal{AD}, m \in \mathcal{M}$, where $\overline{T}_1, \overline{T}_2 \in \mathcal{T}^*$ and $T \neq (\overline{T}_1 \cup \overline{T}_2)$,

$$\mathbf{Pr}[\mathsf{Unwrap}(sk_{\setminus \overline{T}_1}, T, AD, \mathsf{Wrap}(sk_{\setminus \overline{T}_2}, T, AD, m)) = m | sk \leftarrow_{\$} \mathsf{KGen}()] = 1 \quad (1)$$

Similarly, PKW$^+$ requires that the derivation algorithm derives the same key regardless of the punctured state of the secret key sk, as shown in definition 4-(3). sk is the secret key generated by puncturing on $(T_1, ..., T_n \in \mathcal{T})$, where puncturing on SK is not affected by the order of T (PPRF*puncture invariance*) [4]. We also redefine this concept for the PKW$^+$,

Definition 3 (PKW$^+$ puncture invariance). *The* PKW$^+$ *is puncture invariance for all secret keys* $sk \in \mathcal{SK}$ *and all tags* $T_i \in \mathcal{T}$, *where* $i \in \{1, \ldots, n\}$, *and the* sk *is solely influenced by the tags on which punctures have occurred, with no regard for the sequence in which these punctures were executed.*

$$\mathsf{Punc}(\mathsf{Punc}(sk, T_0), T_1) = \mathsf{Punc}(\mathsf{Punc}(sk, T_1), T_0) \quad (2)$$

the key derivation and wrapping remain consistent regardless of the punctured state of the key,

Definition 4 (PKW$^+$ consistency). *The* PKW$^+$ *is consistent if the output of the derivation and wrapping algorithms solely depends on the tag, additional data and the wrapped message but not the punctured state of the secret key unless the output is* $\perp$ *due to puncturing. Thus, if all messages* $m \in \mathcal{M}$, *additional data* $AD \in \mathcal{AD}$ *and all tags* $(T_1, .., T_n) \in \mathcal{T}^*$ *and* $T \in \mathcal{T}$, *where* $T \neq (T_1, .., T_n)$, *it holds that,*

$$\mathbf{Pr}[\mathsf{Drv}(sk, T, AD) = \mathsf{Drv}(sk_{\setminus (T_1, \ldots, T_n)}, T, AD) | sk \leftarrow_{\$} \mathsf{KGen}()] = 1 \quad (3)$$

$$\mathbf{Pr}[\mathsf{Wrap}(sk, T, AD, m) = \mathsf{Wrap}(sk_{\setminus (T_1, \ldots, T_n)}, T, AD, m) | sk \leftarrow_{\$} \mathsf{KGen}()] = 1 \quad (4)$$

5 Adversary Model and Security Goals

In this section, we first define the adversary model and then establish security and privacy goals.

5.1 Adversary Model

Our security model incorporates concepts from both the 3GPP group [1] and recent literature [5, 12, 20] for 5G authentication and handover protocols. Throughout the execution of the proposed scheme, all communication channels between the user and the network are public, signifying that the adversary has complete control over these public channels. For the channels connecting the gNBs and the CN (specifically the N2 interface), TS 33.501 [1] explicitly defines security requirements. These include confidentiality and integrity protection of signalling and user data, as well as authentication between the gNBs and the CN.

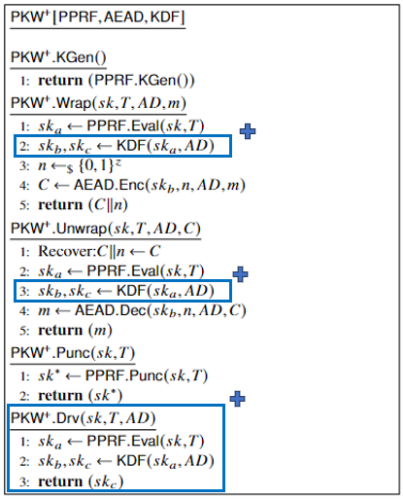


Figure 1: PKW$^+$ Algorithms. Blue boxes represents the modifications PKW$^+$ introduces to PKW .

Our adversary model, illustrated in **Figure 2**, encompasses three distinct types:

(1) $\mathcal{A}_1$: A Dolev-Yao adversary that possesses control over the network, allowing for the interception, insertion, modification, and deletion of any message.
(2) $\mathcal{A}_2$: Seeks to compromise the key-indistinguishability and linkability properties of the communicating parties. It is important to consider this adversary to protect user privacy, prevent tracking, and ensure robust key management.
(3) $\mathcal{A}_3$: Focuses on compromising forward Secrecy and forward privacy. This adversary has the ability to compromise various secret keys, including:
   (a) Long-term keys (LTK) shared between UE and CN
   (b) Asymmetric secret keys (ASK) of the CN
   (c) Session keys established between protocol participants

Protecting against $\mathcal{A}_3$ adversary ensures that protocols can safeguard past communications even if future key compromises occur, maintaining long-term security of the system.

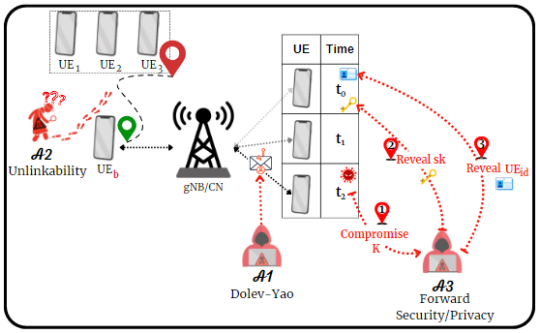


Figure 2: Adversary Model

5.2 Design Goals

To ensure security and privacy in 5G mobile communication, certain requirements must be met. In this context, reviews conducted by [5],[20] and [12] on the 5G-AKA and handover protocols have identified specific security and privacy prerequisites for authentication and handover procedures that require careful consideration. Our security analysis, detailed in Section 7, and the subsequent discussion (covered in Appendix B), ensures that our proposed scheme effectively attains all specified security goals.

(1) ***Mutual authentication*** (MA) is a vital security process that enables two parties engaged in network communication, such as a UE and the CN, to authenticate each other's identities. Ensuring the authenticity of every party in the network is essential to prevent various security threats, including man-in-the-middle attacks.
(2) ***Unlinkability*** (Unlink) in 5G networks is crucial for preserving users privacy, it provides a stronger notion of user privacy in the network compared to a user anonymity alone. User unlinkability is a feature that ensures that the actions or transactions performed by a user within the network cannot be traced back to the same user across different interactions or sessions. In other words, even if a third party observes multiple transactions or activities conducted by a user, they cannot establish a connection or link between these activities to identify the user behind them.
(3) ***Perfect forward secrecy*** (PFS) is a critical security property in 5G networks that ensures the confidentiality of past session keys even if long-term secrets are compromised in the future. This security feature is especially significant in 5G networks, given the frequent handovers resulting from the increased deployment of cells. With numerous session keys generated during each handover, it becomes imperative for 5G protocols to ensure that any compromise does not jeopardise the security of past session keys. In alignment with this imperative, the proposed protocol adheres to the same long-term keys employed in the standard 5G-AKA protocol. However, a key enhancement involves updating this key during every execution of the protocol, with the assistance of PKW^+.
(4) ***Perfect forward Privacy*** (PFP) extends the concept of PFS to the realm of user privacy. While PFS focuses on protecting past session keys, PFP aims to maintain user anonymity and unlinkability even if long-term secret keys are compromised in the future. This property is crucial in the face of increasingly sophisticated adversaries who might gain access to long-term secret keys. Unlike the current 5G-AKA protocol, which is vulnerable to privacy breaches if the long-term secret key (K) is compromised, our proposed scheme leverages PKW^+ to update users' temporary identities during each protocol execution. This approach, combined with ID anonymization, ensures that even if an adversary compromises the long-term secret key, they cannot break user anonymity or link users' temporary identities across sessions.
(5) ***Secure revocation management*** (SRM) is a critical feature in 5G networks that enables efficient and secure removal of users, particularly during handovers. As 5G usage grows, effective subscription management becomes crucial for maintaining network efficiency and security. Our protocol implements a revocation mechanism with a threshold approach, balancing network integrity and operational efficiency. This method ensures prompt disconnection of revoked users, preventing unauthorized access while optimizing resource allocation.

6 The Proposed Scheme

The 5G security architecture relies on the USIM to securely store critical authentication credentials: the long-term key K, SUPI, SQN, and authentication algorithms. Our PGUP scheme extends this established security model by introducing an additional credential, SUTI (Subscription Temporary Identifier), to enhance user privacy. This section introduces the PGUP scheme, which consists of two main protocols: the *initial authentication-AKA* protocol and the *Handover* protocol. We provide the notation used in this paper in Table 2.

6.1 Initial Authentication

All registered users in the network seeking secure access are required to execute this protocol. While we adhere to a similar registration procedure as the current 5G-AKA, our approach differs in the method of SUPI (Subscription Permanent Identifier) protection. In 5G-AKA, this protection is achieved through encrypting the SUPI. In our protocol, we introduce SUTI (Subscription Temporary Identifier), an independent value from SUPI, which provides an extra layer of user identity protection. SUTI is a temporary random identifier generated by the CN and maintained by both the CN and the UE. This approach enhances privacy by further anonymizing the user's permanent identity, going beyond the protection offered in standard 5G-AKA. Following the 5G specifications (TS 33.501) [1], we assume that the public key of the CN is preserved in the USIM. This assumption is crucial for the security of the authentication process. Below, we present a detailed description of the sequence of messages essential for the initial authentication protocol, as illustrated in Figure 3.

Step 1: UE $\rightarrow$ CN : $[\mathrm{SUCI}, ID_{\mathrm{CN}}]$
The proposed AKA follows a similar pattern to the conventional 5G-AKA in its initial steps. It begins by using the same ECIES-encapsulation algorithm to generate keys (ck, C_0). Subsequently, the User Equipment (UE) introduces a random salt (Δ) to update its Subscription Temporary Identity (SUTI). This update is achieved by computing the hash of SUTI concatenated with Δ, resulting in SUTI*. The UE then encrypts the original SUTI using the key ck, producing ciphertext C_1. Next, the UE generates a tag ($T \leftarrow_{\$} \mathcal{T}$) and wrap the new tempoanary identity (SUTI*) using (PKW^+.Wrap) algorithm with the long-term key K, a tag T, and additional data ($AD \leftarrow (C_0)\|C_1)$) to form C_2. Next, we derive a key (rk) using the (PKW^+.Drv) algorithm and update the long-term key by puncturing it on the tag T. Finally, compose a SUCI which consists of (C_2)and send them along with the ID of the supported CN to the gNB, which he will forward to the specified CN.

Step 2: CN → gNB : $[R^{'}, AUTN, HxRES^{*}, K_{SEAF}, HK]$
In this step, the process closely resembles the conventional 5G-AKA, wherein the CN employs the ECIES-decapsulation algorithm to generate ck, mirroring the UE's action. This key is then utilised to decrypt C_1, facilitating the retrieval of (SUTI). The SUTI serves as an index, allowing the CN to identify and access the corresponding long-term key K. The CN derives rk using $\mathsf{PKW}^{+}.\mathsf{Drv}(K, T, C_0 \| C_1)$ and unwraps C_2 to obtain $\mathrm{SUTI}^{*} \| \Delta$. Then, the CN performs a key update (K^{*}) by puncturing on K using the tag T. The CN checks if SHA256(SUTI, Δ) equals SUTI^{*} and updates SUTI to SUTI^{*}. Following this, the CN randomly selects R and encrypts it using ck with Φ and AUTN, where Φ is a fixed string of all zeros used in the protocol. The subsequent steps closely mirror those of the conventional 5G-AKA, incorporating minor adjustments to generate secure values. By utilising the f_5 function, the CN derives the authentication key (AK) and MAC key (MK). Subsequently, employing f_3, the CN generates the session key (SK) and response key (RK). Further, the MK is utilised in f_1 to generate the Message Authentication Code (MAC). The CN then obfuscates the sequence number through an XOR operation with AK. Subsequently, the CN employs the KDF to generate the expected response ($xRES$) and hashes it using SHA256. Crucially, the CN generates both the session key and handover key, denoted as k_{SEAF} and HK, respectively, facilitating communication between the gNB and the user. Next, the CN increments the sequence number (SQN). Finally, the CN sends $(R^{'}, AUTN, HxRES^{*}, K_{SEAF}, HK)$ to the gNB, which he/she forwards $(R^{'}, AUTN)$ only to the UE.
Step 3: UE → gNB : $[RES^{*}]$
Upon message reception, the UE decrypts $R^{'}$ and generates authentication and MAC keys (AK, MK) using the received R and rk. Extracting ($xCONS, XMAC$) from the Authentication Token ($AUTN$). UE deconceals ($xCONC$) to retrieve (SQN_{CN}) through XOR operations. Next, UE uses MK to generate a MAC for the $xSQN_{\mathrm{CN}}, R$ and $SUCI$ using (f_1) function. Then UE compares the generated MAC with $XMAC$ and SQN_{CN} with SQN_{UE}, if both verifications hold UE computes the response key RK and session key SK using the authentication key ($f_3(AK, \mathrm{SUTI})$). Then UE generate another set of keys, K_{SEAF} and HK keys for future communications and handover protocol.
Step 4: gNB → CN : $[RES^{*}, SUCI]$
Upon receiving RES^{*}, the gNB compares $HxRES$ with the hash value of RES^{*}. If both hash values are equal, the gNB sends RES^{*} together with the corresponding $SUCI$ to the CN. The rest of the protocol then proceeds according to 5G-AKA.

Remark 1. *We consistently update the long-term key (K) shared between the* UE *and the* CN *during each execution. These updates are achieved through the* Punc() *algorithm, which serves two critical purposes. Firstly, it ensures PFS, guaranteeing that the confidentiality of past communications is maintained even if the current key is compromised. Secondly, it significantly mitigates risks associated with key-synchronization issues between the* UE *and* CN*. This mitigation is effective due to the sequence independence of the* Punc() *algorithm: the order of punctures does not affect the outcome. This property ensures consistent key derivation regardless of potential message reordering or loss, further strengthening the robustness of the key update mechanism.*

Table 2: Notation

Parameter	Content/Description
R	128 bit Random Number
SQN	48 bit Sequence Number
AK,MK	$f_5(rk, R)$
RK,SK	$f_3(AK, SUTI)$
MAC	$f_1(MK, SQN_{\mathrm{CN}} \| R \| SUCI)$
CONC	$SQN_{\mathrm{CN}} \oplus AK$
AUTN	$< CONC, MAC >$
SUTI	Subscription Temporary Identifier
$xRES^{*}$	$KDF(RK, R \| \mathrm{gNB}_{name})$
$HxRES^{*}$	$SHA256(R \| xRES^{*})$
HK, K_{SEAF}	$KDF(SK; R \| \mathrm{gNB}_{name} \| SQN_{\mathrm{CN}})$
ek, tk	$f_5(HK, \theta)$
HK^{*}	$f_3(tk, r)$
$ck, \widetilde{HK}$	$f_5(HK^{*}, t)$
ck, C_0	ECIES public and private keys
T	a tag $T \in \mathcal{T}$

6.1.1 Integration of PGUP AKA Protocol with 5G-AKA. It's important to highlight that the PGUP AKA protocol follows the same message call flow as the current 5G-AKA protocol: identical messages delivered in the same sequence. The only difference is the content of these messages. For network deployment, 5G can leverage Network Function Virtualization (NFV) capabilities, enabling software-based updates to network components while minimising hardware modifications for both base stations and UEs. This alignment demonstrates the seamless integration of the PGUP, AKA protocol, into the 5G environment.

6.2 Universal Handover (HO)

This protocol is triggered when a UE's current base station (SgNB) can no longer serve it, or when a more suitable base station (TgNB) is discovered. In such cases, the SgNB executes the process of handing over the UE. Below, we present a detailed description of the sequence of messages essential for the HO protocol, as illustrated in Figure 4.

Step 1: SgNB → TgNB/UE : $[C_{BS}, C, a] / [C_{\mathrm{UE}}, r]$
This step begins when SgNB sends a HO request and notification to TgNB and UE, respectively. First, SgNB generates encryption and temporary keys (ek & tk) from a nonce θ and HK (generated during the initial authentication protocol) using $f_5(\cdot)$. Next, SgNB randomly samples (r, a) and then uses the temporary key tk and r to update the HK using $f_3(\cdot)$. Additionally, SgNB generates a session key sk using $KeyGen()$ algorithm. These keys encrypt all user information maintained by the SgNB and are used to send the ciphertext to the UE and TgNB. Specifically, the encryption key (ek) is used to encrypt $SUCI|SQN_{HO}$ using $AEAD.ENC(\cdot)$, generating C_{UE}, where H is a protocol-specific header containing fixed information or a predefined string. Whereas, the session key (sk) is used to encrypt $SUCI|SQN_{HO}|HK^{*}$, generating C_{BS}. SgNB, using the Base Key (BK) shared between base stations beforehand, then wraps the session key using $PKW.Wrap(BK, T, a, sk)$, generating C. Finally, SgNB punctures BK on a tag T and sends the HO request (C_{BS}, C) to the TgNB and HO notification (C_{UE}) to the UE.

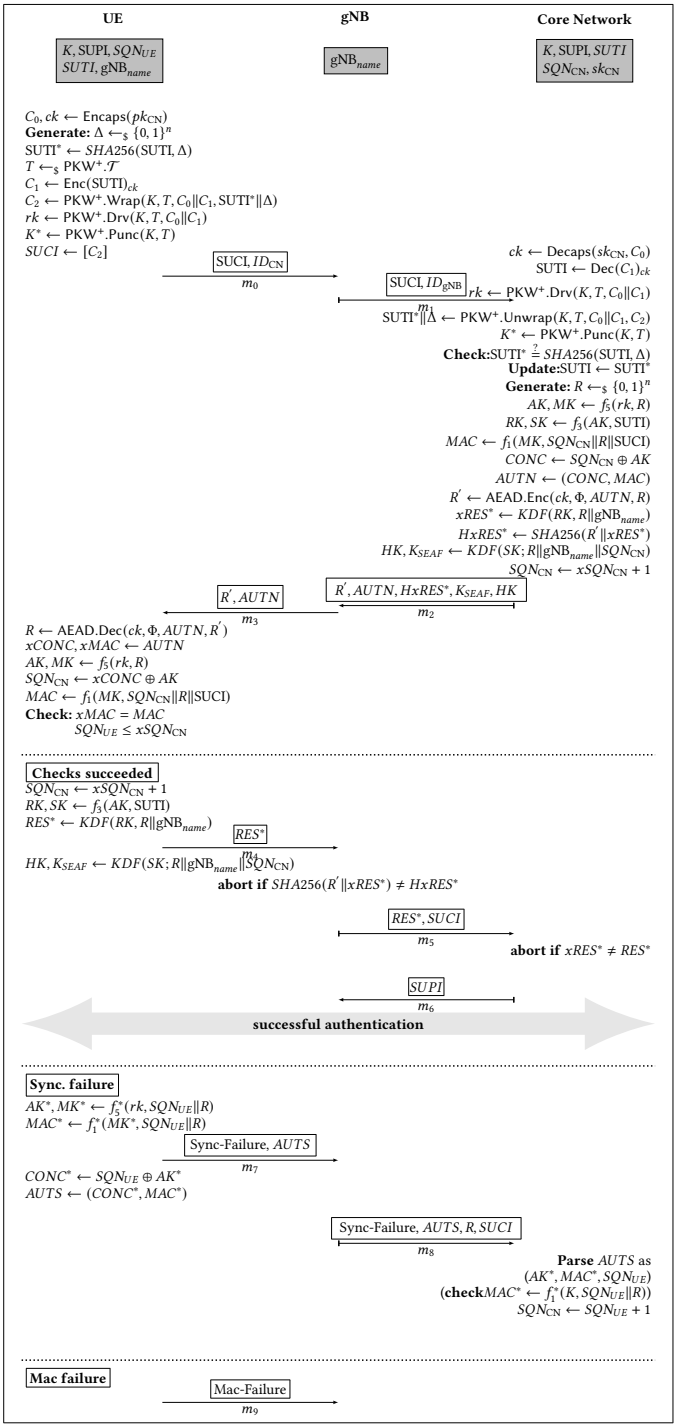


Figure 3: PGUP AKA Protocol

Step 2: TgNB → UE : $[HO_{Res}]$
Upon receiving the HO request (C_{BS}, C) from the SgNB, TgNB retrieves sk by unwrapping C using BK. TgNB then uses sk to decrypt C_{BS} and retrieve the UE's information with the updated handover key $(SUCI \| SQN_{HO} \| HK^*)$. Next, TgNB punctures BK on tag T and checks if SQN_{HO} is less than the threshold n. If the condition holds, TgNB increments SQN_{HO} by one and then randomly samples t to update the handover key HK, generating ck and $\widehat{HK}$. The ck is then used to encrypt the UE's information using the AEAD encryption algorithm ($AEAD.\text{Enc}(ck, t, H, SUCI \| SQN_{HO})$).
Step 3: Upon receiving the HO notification (C_{UE}) from the SgNB, UE generates encryption and temporary keys ($ek\&tk$) from a fixed nonce θ and HK (generated during the initial authentication protocol) using $f_5(\cdot)$. Then UE uses the temporary key tk and r to update the HK using $f_3(\cdot)$. Next, UE decrypts C_{UE} using ek and checks if his information matches and the SQN_{HO} is less than the threshold n.
Step 4: Upon receiving the HO response from TgNB, the UE updates the handover key HK^*, generating ck and $\widehat{HK}$. The ck is then used to decrypt HO_{Res} using the AEAD decryption algorithm. The UE then checks if the decrypted information matches its own records.
Step 5: **Threshold** $[SQN_{HO} \geq n]$ In case the roaming user exceeds the handover threshold (i.e., a user has executed the HO protocol y times, where $SQN_{HO} = y$ and $y \geq n$), the UE resets the handover sequence to zero, setting $SQN_{HO} = 0$. Similarly, TgNB also resets SQN_{HO}, but first checks with the CN if the user is revoked. If the user is revoked, Steps 2 and 4 will not be executed, and the session will be aborted.

6.2.1 Integration of PGUP HO Protocol with 5G-AKA. Here, we demonstrate the seamless integration of the PGUP HO protocol into the existing 5G-HO protocol, as illustrated in Figure 5. The call flow depicted in blue font in Figure 5 outlines precisely where PGUP HO messages can be incorporated without necessitating any infrastructural adjustments. It's worth noting that the figure does not include revocation messages from the PGUP HO protocol, as 5G currently lacks efficient handover support. Unless the 5G group deliberates about incorporating revocation checks, our revocation mechanism cannot be deployed. However, aside from this consideration, the PGUP HO protocol can be effortlessly integrated into the current 5G-HO protocol without requiring any additional infrastructure or modifications.

7 Security Framework and Analysis

In this section, we formalize the security features of the proposed PGUP scheme, which aligns with the key exchange models presented by Bellare-Rogaway [7]. These models essentially represent the security of a key exchange protocol through a game played between a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and a challenger C. The adversary succeeds in the game by causing a winning event (such as breaking authentication or anonymity) or by terminating and correctly guessing a challenge bit b (resulting in breaking key indistinguishability). To articulate notions of *user unlinkability*, we employ the framework developed by Khan et al. [15], while the eCK framework [18] is utilized to capture key indistinguishability (KIND).

7.1 Execution Environment

Here, we outline the common execution environment for all security games. Our analysis involves three distinct games that evaluate

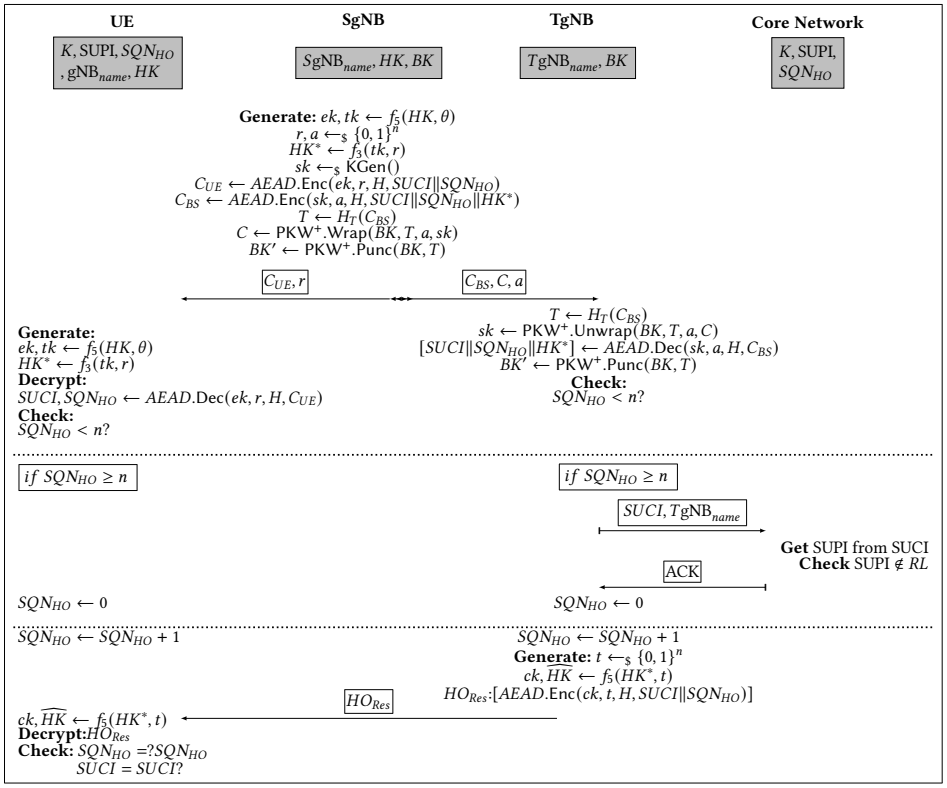


Figure 4: PGUP Handover protocol

various aspects of a key exchange protocol, including mutual authentication, key indistinguishability, and unlinkability.

In these games, the challenger C oversees a singular CN, orchestrating multiple instances of the key exchange protocol Π, as well as (up to) n_P users $\text{UE}_1, \ldots, \text{UE}_{n_P}$ (representing users engaging with the CN) and systems $\text{gNB}_1, \ldots, \text{gNB}_{n_P}$ (representing gNBs interacting with the CN). Each user and system may potentially run up to n_S executions of protocol Π. We simplify notation by using π_i^s to denote the s-th session owned by party i and as a reference to the state maintained by that session. Furthermore, we introduce the state maintained by each session:

- $id \in \{1, ..., n_P\}$: Index of the session owner.
- $\rho \in \{\text{UE}, \text{gNB}, \text{CN}\}$: Role of the session.
- $s \in \{1, ..., n_S\}$: Index of the session.
- $sid \in \{\{0,1\}^*, \perp\}$: Session identifier, initialised as $\perp$.
- $pid \in \{1, ..., n_P, \perp\}$: Partner UE identifier ($\perp$ if $\rho = \text{UE}$).
- $gid \in \{1, ..., n_P, \perp\}$: Partner gNB's identifier.
- $msg_s^r \in \{\{0,1\}^*, \perp\}$: Messages sent by the session to the partner session with role ρ.
- $msg_r^\rho \in \{\{0,1\}^*, \perp\}$: Messages received by the session from the partner session.
- $k_i \in \{\{0,1\}^\lambda, \perp\}$: Long-term CN/UE symmetric-key.
- $k \in \{\{0,1\}^\lambda, \perp\}$: Established session key.
- $\alpha \in \{\texttt{in-progress}, \texttt{accepted}, \perp\}$: Session status.
- $it \in \{\{0,1\}^*, \perp\}$: Secret internal state of the session.

The challenger, generating long-term symmetric and asymmetric-keys for each party, samples a bit b randomly. Now $\mathcal{A}$ has the ability

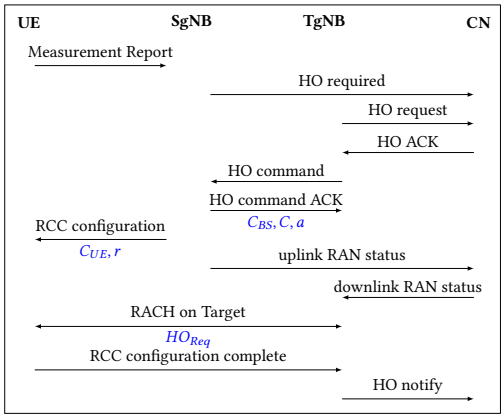


Figure 5: PGUP 5G-HO compatibility

to interact with C through adversary queries. The adversary has the capability to inject, modify, drop, delay, or delete messages as desired, accomplished through **Send** queries. Our models empower $\mathcal{A}$ to initiate UE and gNB sessions owned by specific parties. Additionally, $\mathcal{A}$ has the authority to compromise the long-term secrets of sessions using Corrupt queries, session keys through **Reveal** queries, and the internal state of sessions via **StateReveal** queries, as detailed below.

Adversarial queries. Here we describe queries that enable adversary $\mathcal{A}$ to interact with C throughout the execution of the experiments. The availability of these queries to the adversary may vary across different security games:

- **Create**(i, s, ρ): initiates a new UE or gNB session (denoted π_i^s) such that $\pi_i^s.id = i$ and $\pi_i^s.\rho = \rho$.
- **Send**$(m, i, s, \rho) \rightarrow m'$: transmits message m to π_i^s where $\pi_i^s.\rho = \rho$. π_i^s processes m and may output a new message m'.
- **CorruptLTK**$(i) \rightarrow k_i$: exposes the current shared long-term key K of UE_i and CN.
- **CorruptASK**$(i, \rho) \rightarrow sk$: exposes the long-term asymmetric keys of party $\rho = \text{CN}$ (if $\rho = \text{UE/gNB}$ return $\perp$), returning sk_{CN}.
- **CorruptBK**(i, j): exposes the current shared key BK of SgNB and TgNB.
- **StateReveal**$(i, s, \rho) \rightarrow \pi_i^s$: reveals the internal state of π_i^s where $\pi_i^s.\rho = \rho$.
- **Reveal**$(i, s, \rho) \rightarrow k$: reveals the session key k computed during session π_i^s where $\pi_i^s.\rho = \rho$.
- **Test**$(i, s, \rho) \rightarrow k_b$ (**Exclusive to the** KIND **security experiment**): If **Test** has been previously issued, $\pi_i^s.\alpha = \texttt{accepted}$, or if it does not satisfy the cleanliness predicate $\textbf{clean}^{\pi_i^s}$, as in Definition 10, then C returns $\perp$. Otherwise, C sets $k_0 \leftarrow \pi_i^s.k$, and $k_1 \leftarrow_\$ \{0,1\}^\lambda$, subsequently returning k_b to $\mathcal{A}$ (where b was sampled by C at the beginning of the experiment).
- **Test**$(s, i, s', i') \rightarrow m$ (**Exclusive to the** Unlink **security experiment**): Empowers $\mathcal{A}$ to participate in the Unlink security game. Upon receiving a **Test**(s, i, s', i') query, C initializes a new session π_b (where $\pi_0 = \pi_i^s$ and $\pi_1 = \pi_{i'}^{s'}$), and both $[\textbf{clean}^{\pi_i^s} = \textbf{clean}^{\pi_{i'}^{s'}} = \mathit{true}]$ (see Definition 11). The issuance of a **Test** query by $\mathcal{A}$ is only permissible if there is no session $\pi.\alpha \neq \texttt{in-progress}$ where $\pi.id = i/i'$. C will respond to any **Send**$(m, i,$ $s, \text{UE})$ or **Send**(m, i', s', UE) queries with $\perp$ until $\pi_b.\alpha \neq \texttt{in-progress}$.
- **SendTest**$(m) \rightarrow (m')$ (**Exclusive to the** Unlink **security experiment**): Empowers $\mathcal{A}$ to transmit a message m to π_b subsequent to issuing **Test**. C responds with a $\perp$ if $\pi_b.\alpha \neq \texttt{in-progress}$.

7.2 Matching Conversations

To determine which secrets the adversary can compromise without trivially breaking the security of our scheme, it is crucial to define how sessions are *partnered* and whether those sessions meet the criteria of being *clean*. Fundamentally, partnering ensures that we can trace significant sessions back to other corruptions induced by $\mathcal{A}$, while cleanness predicates establish which secrets $\mathcal{A}$ is not authorised to compromise. Traditionally, matching conversations are employed in the BR model [6], and the eCK-PFS model refines this concept to *origin sessions*. However, these partnering methods prove insufficient for our scenario, where the gNB essentially relays messages between the UE and the CN. Consequently, two challenges arise: we must capture the messages that UE authenticates to the CN, and we also need to address the fact that the gNB sends messages to two parties, neither of which precisely matches the gNB's transcript. To tackle these challenges, our approach is to redefine the matching conversation, which we call it *matching subsets* to handle our scenario better and to encompass the subset of messages authenticated between the gNB and both the CN and the UE.

Definition 5 (Matching Subset). *Consider $S \subseteq T$ to denote that all strings s in the set S are substrings of T. A session π_i^s is deemed to have a matching subset with another session π_j^t under the following conditions:*

- *$(\pi_i^s.msg_r^\sigma \subseteq \pi_j^t.msg_s^\tau)$, $(\pi_i^s.\rho \neq \pi_j^t.\rho)$ and $(\pi_i^s.\rho = \tau, \pi_j^t.\rho = \sigma)$*
- *$(\pi_j^t.msg_r^\tau \subseteq \pi_i^s.msg_s^\sigma)$*
- *$(\pi_j^t.msg_r^\tau \neq \perp)$ or $(\pi_i^s.msg_r^\sigma \neq \perp)$*

We denote sessions with matching subsets as matching sessions.

7.3 Mutual Authentication(MA-Security)

In this section, we describe the main objective of $\mathcal{A}$ in the MA security game and explain the queries available to $\mathcal{A}$. The experiment $\textbf{Exp}^{\text{MA,clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ unfolds in the interaction between a challenger C and an adversary $\mathcal{A}$. Commencing the experiment, C generates long-term asymmetric keys for the CN, alongside long-term symmetric keys for each user UE_i (where $i \in [n_P]$), and sampling a random bit b. Subsequently, C interacts with $\mathcal{A}$ through queries such as **Create**, **Send**, **CorruptLTK**, **CorruptASK**, **StateReveal**, and **Reveal**. $\mathcal{A}$ wins (resulting in $\textbf{Exp}^{\text{MA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ outputting 1) if the adversary induces a **clean** session to accept (setting $\pi_i^s.\alpha \leftarrow \texttt{accepted}$) and if no matching subset session π_j^t exists.

We now delve into the description of our cleanness predicates. In the initial authentication protocol, if a session (owned by CN or UE) accepts without a matching subset, $\mathcal{A}$ wins only if they haven't compromised: The state of any matches, or both the long-term shared key of the UE partner and the long-term asymmetric-key of the CN.

Definition 6 (Initial authentication cleanness). *A session π_i^s in the* MA *experiment described above is* $\textbf{clean}_{IA}$ *if the following conditions hold:*

(1) ***StateReveal***$(i, s, \pi_i^s.\rho)$ *has not been issued* and *for all sessions π_j^t such that π_j^t is a matching subset of π_i^s* ***StateReveal***$(j, t, \pi_j^t.\rho)$ *has not been issued.*

(2) If $(\pi_i^s.\rho = \text{gNB})$ or $(\pi_i^s.\rho = \text{CN})$, and there exists no $(j, t) \in n_P \times n_S$ such that π_j^t is a matching session of π_i^s, ***CorruptLTK***$(\pi_i^s.pid)$ *have not been issued before $\pi_i^s.\alpha = \texttt{accepted}$.*

(3) If $\pi_i^s.\rho = \text{UE}$, and there exists no $(j, t) \in n_P \times n_S$ such that π_j^t is a matching session of π_i^s, ***CorruptLTK***(i) *or* ***CorruptASK***(CN) *have not both been issued before $\pi_i^s.\alpha = \texttt{accepted}$.*

Definition 7 (Handover cleanness). *A session π_i^s in the* MA *experiment described above is* $\textbf{clean}_H$ *if the following conditions hold:*

(1) ***StateReveal***$(i, s, \pi_i^s.\rho)$ *has not been issued* and *for all sessions π_j^t such that π_j^t is a matching subset of π_i^s* ***StateReveal***$(j, t, \pi_j^t.\rho)$ *has not been issued.*

We assert that a protocol Π is considered MA-secure if no probabilistic polynomial-time (PPT) algorithms $\mathcal{A}$ capable of winning the MA security game against a clean session with a non-negligible advantage. We formalise this notion below.

Definition 8 (Mutual Authentication Security). *Let Π be a key exchange protocol, and $n_P, n_S \in \mathbb{N}$. For a given cleanness predicate* **clean**, *and a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the mutual authentication* MA *game to be:* $\textbf{Adv}^{\text{MA,clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\textbf{Exp}^{\text{MA,clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$. *We state that Π is* MA*-secure if, for all PPT $\mathcal{A}$,* $\textbf{Adv}^{\text{MA,clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in security parameter λ.*

Theorem 1. MA***-security of PGUP Initial Authentication****. PGUP AKA protocol depicted in Figure 3 is* MA*-secure under the cleanness predicate* $\textbf{clean}_{IA}$ *in Definition 6. For any PPT algorithm*

$\mathcal{A}$, $\mathbf{Adv}^{\mathsf{MA},\mathbf{clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible assuming the security of* PKW$^+$, AEAD, *MAC, ECIES and* KDF *schemes* [†].

Theorem 2. MA-***security of PGUP Handover***. *The PUGP HO protocol depicted in* ***Figure*** *4 is* MA*-secure under the cleanness predicate in Definition 7. For any PPT algorithm $\mathcal{A}$ against the* MA *experiment,* $\mathbf{Adv}^{\mathsf{MA},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible assuming the security of* PKW$^+$, AEAD, *MAC, ECIES and* KDF *schemes*[†].

7.4 Key Indistinguishability (KIND-Security)

Here, we describe the main objective of $\mathcal{A}$ in the KIND security game and explain the queries available to $\mathcal{A}$. The experiment $\mathbf{Exp}^{\mathsf{KIND},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ unfolds the interactions between a challenger C and an adversary $\mathcal{A}$. At the onset of the experiment, C generates long-term symmetric keys for the CN and each user UE_i and each gNB_i (where $i \in [n_P]$), samples a random bit $b \leftarrow_\$ \{0,1\}$, and subsequently engages with $\mathcal{A}$ through the queries detailed below. At a certain juncture, $\mathcal{A}$ issues a $\mathbf{Test}(i,s,\rho)$ query and either receives $\pi^s_i.k$ or a random key from the same distribution (based on b). $\mathcal{A}$ concludes its interactions, produces a guess b', and secures victory (resulting in $\mathbf{Exp}^{\mathsf{KIND},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ outputting 1) if $b' = b$.

We assert that a protocol Π is considered KIND-secure if there are no probabilistic polynomial-time (PPT) algorithms $\mathcal{A}$ capable of winning the KIND security game against a clean session with a non-negligible advantage. We formalise this notion below.

Definition 9 (Key Indistinguishability). *Let* Π *be a key exchange protocol, and* $n_P, n_S \in \mathbb{N}$. *For a given cleanness predicate* **clean**, *and a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the key indistinguishability* KIND *game to be:* $\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) = |\Pr[\mathbf{Exp}^{\mathsf{KIND},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}|$. *We say that* Π *is* KIND*-secure if, for all PPT $\mathcal{A}$,* $\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ *is negligible in the parameter λ.*

Definition 10 (cleanness predicate). *A session* π^s_i *in the* KIND *experiment described above is* $\mathbf{clean}_{IA}$ *if conditions (1,2,3) hold, and is* $\mathbf{clean}_H$ *if conditions (1,2,4) hold:*

1. $\boldsymbol{Reveal}(i,s,\rho)$ *has not been issued, and if a matching session* π^t_j *exists,* $\boldsymbol{Reveal}(j,t,\pi^t_j.\rho)$ *has not been issued.*
2. *The query* $\boldsymbol{StateReveal}(i,s,\rho)$ *has not been issued* and *for all j,t such that* π^t_j *has a matching conversation with* π^s_i, $\boldsymbol{StateReveal}(j,t,\pi^t_j.\rho)$ *has not been issued.*
3. *If there is no* $(j,t) \in n_P \times n_S$ *such that* π^t_j *is a matching subset for* π^s_i, $\boldsymbol{CorruptLTK}(i)$ *and* $\boldsymbol{CorruptASK}(\pi^s_i.pid)$ *have not been both issued before* $\pi^s_i.\alpha = \texttt{accepted}$.
4. $\boldsymbol{CorruptBK}(i,j)$ *has not been issued before* $\pi^s_i.\alpha = \texttt{accepted}$.

Theorem 3. KIND-***security of the PGUP scheme***. *The proposed scheme, as shown in* ***Figures*** *3 and 4, are considered* KIND*-secure and supports PFS, provided that it satisfies the cleanness predicate described in Definition 10. In other words, for any PPT algorithm $\mathcal{A}$ attempting to breach the* KIND *experiment, the probability of success is negligible. This holds true under the assumption* PKW$^+$, AEAD, *MAC, ECIES and* KDF *security are all maintained*[†].

[†]**The security analysis of all theorems are provided in the appendices**

7.5 Unlinkability (Unlink-Security)

Here, we describe the main objective of $\mathcal{A}$ in the Unlink security game and explain the queries available to $\mathcal{A}$. The experiment $\mathbf{Exp}^{\mathsf{Unlink},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ unfolds in an interaction between a challenger C and an adversary $\mathcal{A}$. Commencing the experiment, C generates symmetric keys for the CN as well as for each user UE_i and each gNB_i (where $i \in [n_P]$). Subsequently, a random bit b is sampled from $\{0,1\}$, and C engages with $\mathcal{A}$ through the specified queries below. The $\mathcal{A}$ wins (resulting in $\mathbf{Exp}^{\mathsf{Unlink},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ outputting 1) when it successfully concludes, producing a guessed bit b' that aligns with the initially chosen bit b.

Adversary Queries In this Unlink game, we consider the $\mathcal{A}$ has access to several queries including **Create**, **Send**, **CorruptLTK**, **CorruptASK**, **Reveal**, **StateReveal**, **Test**, and **SendTest**. In contrast to the KIND game, the **Test** query in the Unlink scenario enables the adversary to initiate one of two sessions based on a bit b randomly chosen by the challenger. Additionally, **SendTest** allows the adversary to engage with that session without disclosing its ownership. We now proceed to define our cleanliness predicate for the unlinkability game. This predicate is identical to the $\mathbf{clean}_{IA}$, as previously defined in 6.

Definition 11 (Cleanness predicate). *A session* π^s_i *in the* Unlink *experiment is* **clean** *if the following conditions hold:*

1. $\boldsymbol{StateReveal}(i,s,\pi^s_i.\rho)$ *has not been issued* and *for all sessions* π^t_j *such that* π^t_j *is a matching subset of* π^s_i $\boldsymbol{StateReveal}(j,t,\pi^t_j.\rho)$ *has not been issued.*
2. *If* $(\pi^s_i.\rho = \mathrm{gNB})$ *or* $(\pi^s_i.\rho = \mathrm{CN})$, *and there exists no* $(j,t) \in n_P \times n_S$ *such that* π^t_j *is a matching session of* π^s_i, $\boldsymbol{CorruptLTK}(\pi^s_i.pid)$ *have not been issued before* $\pi^s_i.\alpha = \texttt{accepted}$.
3. *If* $\pi^s_i.\rho = \mathrm{UE}$, *and there exists no* $(j,t) \in n_P \times n_S$ *such that* π^t_j *is a matching session of* π^s_i, $\boldsymbol{CorruptLTK}(i)$ *or* $\boldsymbol{CorruptASK}(\mathrm{CN})$ *have not both been issued before* $\pi^s_i.\alpha = \texttt{accepted}$.

Theorem 4. Unlink-***security of the PGUP scheme***. *The proposed scheme, as shown in* ***Figures*** *3 and 4, are considered* Unlink*-secure and supports PFP, provided that it satisfies the cleanness predicate described in Definition 11. In other words, for any PPT algorithm $\mathcal{A}$ attempting to breach the* Unlink *experiment, the probability of success is negligible. This holds true under the assumption* PKW$^+$, AEAD, *MAC, ECIES and* KDF *security are all maintained*[†].

8 Evaluation and Comparison

In this section, we provide a comprehensive performance analysis of the proposed PGUP scheme. The evaluation is divided into three main components. First, we analyze the computational overhead of the cryptographic operations used to design our proposed protocol and assess their combined impact on the overall performance of the scheme. Next, we evaluate the PGUP scheme's performance through real-time implementation simulations using the OpenAir-Interface framework. Finally, we assess the proposed PKW$^+$ scheme and conduct a comparative analysis with the original PKW implementation.

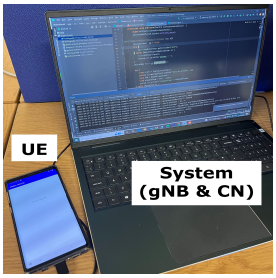


Figure 6: PGUP Testbed

8.1 Evaluation of Cryptographic Operations on Phone-based Testbed

8.1.1 ***Experimental Setup.*** To evaluate our PGUP scheme against conventional 5G-AKA and HO protocols, we established a testbed, as shown in Figure 6, simulating network entities (CN, gNB). We used a Dell Inspiron (i7 core, 2.30GHz CPU, 16.0 GB RAM) for the network entities and an Android-10 smartphone (octa-core 1.8GHz Quad-Core ARM Cortex-A55, 2.7GHz Quad-Core Mongoose M3, 6GB RAM) for the UE. Cryptographic operations were implemented using Java Cryptography Extension (JCE) [25]. We analyzed both computational and communication costs, considering factors such as propagation and transmission time, message dimensions, and network data rates. Following 3GPP specifications [1], we used 25 Mbps uplink and 50 Mbps downlink rates for a wide-area scenario. Propagation delay was calculated using a wave speed of $3 \times 10^8 m/s$ and a maximum 5G cell size of 200 m, resulting in a 0.67 μs delay.

To measure the communication cost for the PGUP scheme, we aligned cryptographic element sizes with 5G specifications where possible, balancing security and efficiency. The SUTI is 8 bytes, while its hashed version, SUTI*, is 32 bytes, consistent with the SHA-256 output. The Δ parameter, occupying 4 bytes, provides anonymization for the SUTI. The puncturable key wrapping tag is 16 bytes. For ECDH key exchange (used in ECIES), C_0 (public key) is 91 bytes, typical for a 256-bit ECDH key. The C_1 parameter is 8 bytes, while C_2 totals 163 bytes, comprising 64 bytes of ciphertext (12-byte nonce, 36-byte encrypted message, 16-byte authentication tag) and 99 bytes of associated data. Adhering to 5G specifications, AUTN, RES, and R are each 16 bytes. R', another ciphertext with AD, is 32 bytes. Both C_{UE} and HO_{res} mirror C_2 at 163 bytes. These carefully chosen sizes balance security requirements, 5G compatibility, and efficient communication in our PGUP scheme.

8.1.2 ***Performance Results.*** Table 3 shows the mean time, based on 100 experimental runs, required for executing the initial authentication and HO protocols, capturing the total time required for protocol execution at both the User Equipment level (T_{UE}) and the system level (T_{Sys}) (i.e., gNB and CN). In terms of computation, PGUP's AKA protocol shows increased computational costs compared to Conventional-5G, it offers improvements in handover operations. PGUP's HO protocol notably demonstrates an 85.5% reduction in UE-level computational cost (0.342 ms vs 2.366 ms for Conventional-5G). This substantial reduction is achieved through optimized cryptographic operations, where PGUP requires fewer and lighter cryptographic computations at the UE side compared to the multiple encryption and decryption cryptographic operations needed in conventional 5G-HO. This improvement particularly benefits user equipment with limited resources, such as processing power and memory. In terms of communication costs, PGUP's HO protocol achieves a 36.5% reduction in total communication cost (0.0535 ms vs 0.08425 ms for Conventional-5G). A key advantage of PGUP's HO protocol is eliminating uplink communication costs, which can significantly reduce network load during handover operations. The confidence intervals, represented as mean ± margin of error in Table 3, for the computation costs in the PGUP and 5G-AKA HO protocols were calculated based on 100 experimental runs at a 95% confidence level. The results indicate reliable precision and controlled variability across experiments, suggesting consistent performance in the PGUP scheme. For example, the confidence interval for T_{UE} in the PGUP AKA scheme was [6.94, 7.08] ms, and for T_{Sys} in the PGUP HO scheme, it was [4.76, 4.86] ms.

Table 3: Phone-based performance comparison.

Schemes \ Cost	Entity	Comp.cost(ms) AKA	Comp.cost(ms) HO	Link	Comm.cost(ms) AKA	Comm.cost(ms) HO
Conventional-5G [1]	T_{UE}	2.030 (±0.004)	2.366 (±0.004)	UP	0.02764	0.04358
	T_{Sys}	1.091 (±0.003)	1.640(±0.004)	Down	0.00646	0.04067
	Total	3.12	3.97	Total	0.0341	0.08425
PGUP	T_{UE}	7.01 (±0.07)	0.342 (±0.002)	UP	0.05862	0
	T_{Sys}	3.24 (±0.03)	4.810 (±0.050)	Down	0.008349	0.0535
	Total	10.24	5.152	Total	0.066969	0.0535

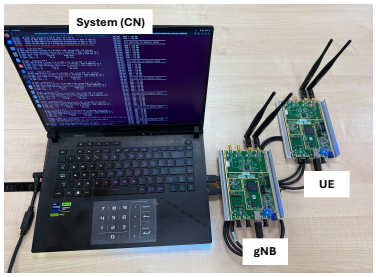
UE: User Equipment, **Sys**:System

Next, we analyzed the battery consumption of the UE for both the Conventional 5G and PGUP schemes. The communication energy consumption for both schemes in the AKA protocol was similar, as PGUP follows the same message flow and message sizes as 5G AKA, resulting in a total of 131.6 μJ for transmitting m_0 (SUCI, 163 bytes) and receiving m_3 (R + AUTN, 32 bytes) and m_4 (RES, 16 bytes). For the HO protocol, the PGUP scheme requires 114.8 μJ for receiving two messages of 163 bytes each, C_{UE} and HO_{Res}. For computation, the energy consumption differs between the two schemes. In the Conventional-5G scheme, the AKA protocol required 1.55 mJ, and the HO protocol consumed 1.80 mJ. In contrast, the PGUP scheme consumed more energy in the AKA protocol (5.34 mJ) due to additional security features but required significantly less energy for the HO protocol (0.26 mJ), demonstrating a clear optimization for frequent handovers. These results highlight a trade-off in PGUP: while it incurs a higher initial computational and communication cost during AKA, it reduces the energy burden during subsequent HO processes, making it suitable for dynamic 5G environments. These results suggest that PGUP offers substantial improvements in handover efficiency, particularly beneficial for UE performance and network load management during frequent handovers in 5G networks. The increased computational cost for AKA may be justified by enhanced security features, making PGUP a viable option for scenarios requiring robust protection against emerging threats in 5G networks.

8.2 Evaluation Results with SDR-based Testbed

In this section, we provide the details of the simulation analysis based on the real-world 5G testbed.

8.2.1 ***System Configuration.*** As shown in Figure 7, to implement a proof of concept (PoC) and evaluate the proposed PGUP scheme, we constructed a comprehensive testbed environment utilizing an ASUS laptop with an i9 processor and 16 GB RAM,




```
[NAS]   Generating PGUP Message m0.
[NAS]   Generating SUPI.
[NAS]   Generating SUPI Successfully.
[NAS]   Generating data using PKW+.
[NAS]   Generating PGUP Message m0 Successfully.
```

Figure 7: (Top) 5G Testbed Setup, and (Bottom) Log From UE.

operating on Ubuntu 22.02. This setup includes two USRP B210 software-defined radios (SDRs) interfaced with the laptop. We use the OpenAirInterface (OAI) 5G stack, a widely used open-source platform, to establish a 5G network comprising the UE, 5G gNB, and core network components. To implement our customized authentication protocol within the 5G architecture, we extended and modified the OAI code across the core network, gNB, and UE stacks*. In this configuration, one USRP operates as the gNB, while the second USRP emulates a UE, both connected to the same laptop.

8.2.2 ***Performance Results.*** In this experiment, we present a comprehensive, end-to-end evaluation of our proposed scheme on the testbed, designed to measure the total latency of the protocol within a real-world 5G environment. The primary objective of this experiment is to assess and compare the latency incurred by our proposed PGUP-AKA scheme with the 5G-AKA protocol. The collected data, summarized in Table 4, highlights the latency differences and performance characteristics observed under real-world conditions. According to Table 4, our proposed PGUP-AKA scheme takes 1.38s while the conventional 5G-AKA takes 1.35s. The results we received were based on 30 experimental runs at a 95% confidence level. We can notice that there are some differences between the SDR-based results and the phone-based results. This is because the end-to-end latency can be decomposed into computational costs (the cryptographic operations), communication costs (data transmission), and all additional delays occurring across the UE, gNB, and CN interfaces.

Table 4: End-to-End SDR-based comparison.

Scheme	PGUP-AKA	Conventional 5G-AKA
End-to-End Time	1.38s	1.35s
Confidence Interval	[1.34, 1.41]	[1.31, 1.39]

8.2.3 ***Statistical Analysis.*** The experimental results indicate that PGUP achieved a mean of 1.38 with a standard deviation of 0.096, while conventional 5G exhibited a mean of 1.35 with a standard deviation of 0.112. To assess the statistical significance of the observed difference, a two-sample t-test assuming unequal variances was conducted, yielding a p-value of 0.328. This result suggests that the difference is not statistically significant under conventional thresholds (e.g., $p > 0.05$). Additionally, the 95% confidence intervals for PGUP and conventional 5G were [1.34, 1.41] and [1.31, 1.39], respectively, showing substantial overlap. These findings indicate no statistically meaningful difference between the two schemes under the current sample size. This clearly demonstrates that the proposed PKW$^+$ introduces minimal computational overhead. In a nutshell, here we argue that our enhanced protocol archives all the objectives as discussed in Section 5.2 with minimal additional cost.

8.3 PKW+ Performance

In addition to comparing PGUP with conventional 5G protocols, we also evaluated the performance of our new PKW$^+$ scheme against the original PKW *. Table 5 shows the execution times for key algorithms in both schemes.

Table 5: Performance comparison of PKW and PKW+

Function	PKW (ms)	PKW+ (ms)	Difference (%)
Wrap(.)	1.190 (±0.006)	1.21(±0.03)	+2.12%
Unwrap(.)	1.14 (±0.03)	1.2 (±0.1)	+8.58%
Punc (.)	2.8 (±0.1)	2.8 (±0.09)	+0.62%
DRV(.)	0	0.005 (±0.001)	N/A
Total	5.08693	5.23248	+2.86%

As shown in Table 5, PKW$^+$ introduces a slight overhead compared to the original PKW, with a total increase in execution time of 2.86%. This minor performance cost is primarily due to the addition of the Drv function and small increases in Wrap and Unwrap times. However, the enhanced security features provided by PKW$^+$, including improved key separation and reduced risks associated with key reuse, justify this modest performance trade-off. Notably, the Punc algorithm has not been modified in PKW$^+$, and its slight increase is not directly related to the new features implemented in PKW$^+$.

9 Conclusion

This paper introduced Pretty Good User Privacy (PGUP), a novel symmetric-based scheme addressing critical security and privacy vulnerabilities in current 5G-AKA and 5G-HO protocols. As the first standalone symmetric-based solution achieving PFS, PFP, user unlinkability, SRM, and seamless UHO, PGUP represents a significant advancement in 5G security. Key contributions include the development of PKW$^+$, a new variant of Puncturable Key Wrapping tailored for 5G, a comprehensive formal security analysis demonstrating PGUP's robustness, and comprehensive performance analysis.

However, there are two limitations of the proposed PGUP scheme. First, the PGUP HO protocol provides additional revocation checks that current 5G networks do not support. Until 5G incorporates efficient handover support and implements revocation checks, our proposed PGUP handover scheme will be challenging to apply in 5G networks. Another limitation of this work is that while PGUP achieves PFP and PFS using symmetric-based cryptography, neither the scheme nor its security model considered post-compromise attacks, offering the potential for future enhancements against post-compromise attacks. Despite these limitations, which we will consider as future work, PGUP's design maintains conceptual alignment with existing 5G protocol structures, facilitating potential integration with minimal modifications to existing infrastructure.

*The source code for PKW$^+$ and OpenAirInterface implementation described in this paper can be accessed from here https://github.com/YYangNUS/PETS_PGUP.

References

[1] ETSI 3rd Generation Partnership Project (3GPP). 2020. *Security architecture and procedures for 5G System.* Technical Specification version 16.3.0 Release 16. 3GPP. https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/16.03.00_60/ts_133501v160300p.pdf

[2] Rabiah Alnashwan, Prosanta Gope, and Benjamin Dowling. 2023. Privacy-Aware Secure Region-Based Handover for Small Cell Networks in 5G-Enabled Mobile Communication. *IEEE Transactions on Information Forensics and Security* 18 (2023), 1898–1913. https://doi.org/10.1109/TIFS.2023.3256703 Conference Name: IEEE Transactions on Information Forensics and Security.

[3] Jari Arkko, Pasi Eronen, Vesa Lehtovirta, and Vesa Torvinen. 2020. Improved Extensible Authentication Protocol Method for 3GPP Mobile Network Authentication and Key Agreement (EAP-AKA. https://tools.ietf.org/html/draft-ietf-emu-rfc5448bis-08

[4] Matilda Backendal, Felix Günther, and Kenneth G. Paterson. 2022. Puncturable Key Wrapping and Its Applications. In *Advances in Cryptology – ASIACRYPT 2022 (Lecture Notes in Computer Science)*, Shweta Agrawal and Dongdai Lin (Eds.). Springer Nature Switzerland, Cham, 651–681. https://doi.org/10.1007/978-3-031-22966-4_22

[5] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A Formal Analysis of 5G Authentication. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Association for Computing Machinery, New York, NY, USA, 1383–1396. https://doi.org/10.1145/3243734.3243846

[6] Mihir Bellare and Chanathip Namprempre. 2008. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *Journal of Cryptology* 21, 4 (Oct. 2008), 469–491. https://doi.org/10.1007/s00145-008-9026-x

[7] Mihir Bellare and Phillip Rogaway. 2006. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In *Advances in Cryptology - EUROCRYPT 2006 (Lecture Notes in Computer Science)*, Serge Vaudenay (Ed.). Springer, Berlin, Heidelberg, 409–426. https://doi.org/10.1007/11761679-25

[8] Dan Boneh and Brent Waters. 2013. Constrained Pseudorandom Functions and Their Applications. In *Advances in Cryptology - ASIACRYPT 2013*, Kazue Sako and Palash Sarkar (Eds.). Springer, Berlin, Heidelberg, 280–300. https://doi.org/10.1007/978-3-642-42045-0_15

[9] Ravishankar Borgaonkar, Lucca Hirschi, Shinjo Park, and Altaf Shaik. 2018. *New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols.* Technical Report 1175. http://eprint.iacr.org/2018/1175

[10] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. 2014. Functional Signatures and Pseudorandom Functions. In *Public-Key Cryptography – PKC 2014*, Hugo Krawczyk (Ed.). Springer, Berlin, Heidelberg, 501–519. https://doi.org/10.1007/978-3-642-54631-0_29

[11] An Braeken. 2020. Symmetric key based 5G AKA authentication protocol satisfying anonymity and unlinkability. *Computer Networks* 181 (Nov. 2020), 107424. https://doi.org/10.1016/j.comnet.2020.107424

[12] Cas Cremers and Martin Dehnel-Wild. 2019. Component-Based Formal Analysis of 5G-AKA: Channel Assumptions and Session Confusion. San Diego, CA, USA. https://publications.cispa.saarland/2758/

[13] Chun-I Fan, Jheng-Jia Huang, Min-Zhe Zhong, Ruei-Hau Hsu, Wen-Tsuen Chen, and Jemin Lee. 2020. ReHand: Secure Region-Based Fast Handover With User Anonymity for Small Cell Networks in Mobile Communications. *IEEE Transactions on Information Forensics and Security* 15 (2020), 927–942. https://doi.org/10.1109/TIFS.2019.2931076

[14] Pierre-Alain Fouque, Cristina Onete, and Benjamin Richard. 2016. *Achieving Better Privacy for the 3GPP AKA Protocol.* Technical Report 480. Darmstadt, Germany. http://eprint.iacr.org/2016/480

[15] Haibat Khan, Benjamin Dowling, and Keith M. Martin. 2018. *Identity Confidentiality in 5G Mobile Telephony Systems.* Technical Report 876. http://eprint.iacr.org/2018/876

[16] Adrien Koutsos. 2019. The 5G-AKA Authentication Protocol Privacy. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. 464–479. https://doi.org/10.1109/EuroSP.2019.00041

[17] Hugo Krawczyk. 2010. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In *Advances in Cryptology – CRYPTO 2010 (Lecture Notes in Computer Science)*, Tal Rabin (Ed.). Springer, Berlin, Heidelberg, 631–648. https://doi.org/10.1007/978-3-642-14623-7_34

[18] Brian LaMacchia, Kristin Lauter, and Anton Mityagin. 2007. Stronger Security of Authenticated Key Exchange. In *Provable Security (Lecture Notes in Computer Science)*, Willy Susilo, Joseph K. Liu, and Yi Mu (Eds.). Springer, Berlin, Heidelberg, 1–16. https://doi.org/10.1007/978-3-540-75670-5

[19] Christopher Patton and Thomas Shrimpton. 2019. Security in the Presence of Key Reuse: Context-Separable Interfaces and their Applications. https://eprint.iacr.org/2019/519 Publication info: A major revision of an IACR publication in CRYPTO 2019.

[20] Aleksi Peltonen, Ralf Sasse, and David Basin. 2021. A comprehensive formal analysis of 5G handover. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '21)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3448300.3467823

[21] Phillip Rogaway. 2002. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and communications security (CCS '02)*. Association for Computing Machinery, New York, NY, USA, 98–107. https://doi.org/10.1145/586110.586125

[22] Phillip Rogaway and Thomas Shrimpton. 2006. A Provable-Security Treatment of the Key-Wrap Problem. In *Advances in Cryptology - EUROCRYPT 2006*, Serge Vaudenay (Ed.). Springer, Berlin, Heidelberg, 373–390. https://doi.org/10.1007/11761679_23

[23] Amit Sahai and Brent Waters. 2014. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC '14)*. Association for Computing Machinery, New York, NY, USA, 475–484. https://doi.org/10.1145/2591796.2591825

[24] Paul Schmitt and Barath Raghavan. 2021. Pretty Good Phone Privacy. 1737–1754. https://www.usenix.org/conference/usenixsecurity21/presentation/schmitt

[25] Oracle Technology Network. 2022. Java Cryptography Architecture (JCA) Reference Guide. https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html

[26] Yuchen Wang, Zhenfeng Zhang, and Yongquan Xie. 2021. {Privacy-Preserving} and {Standard-Compatible} {AKA} Protocol for 5G. 3595–3612. https://www.usenix.org/conference/usenixsecurity21/presentation/wang-yuchen

[27] Hexuan Yu, Changlai Du, Yang Xiao, Angelos Keromytis, Chonggang Wang, Robert Gazda, Y. Hou, and Wenjing Lou. 2024. AAKA: An Anti-Tracking Cellular Authentication Scheme Leveraging Anonymous Credentials. https://doi.org/10.14722/ndss.2024.24617

Appendices

A PKW$^+$ Security

In accordance with the work of Rogaway and Shrimpton [22], we adopt the concept of "DAE security" (deterministic authenticated encryption), which combines confidentiality and integrity notions. They demonstrate that this combined notion is equivalent to two separate notions in their context, as well as in authenticated encryption overall. Here, we extend this finding to the PKW$^+$ context and formally validate that a similar equivalence exists for our forward security notions. Our combined confidentiality and integrity framework, outlined in **Figure** 9, captures $ind\$ - cca$ and KIND. It also ensures forward security, meaning that confidentiality assurances persist even after compromising the secret key, provided it has been appropriately punctured before corruption to prevent trivial wins. Our combined security notion (confidentiality and integrity) is tailored to the PKW$^+$ setting. In-game $\mathbf{G}^{\mathbf{ind\$-cca}}_{\mathrm{PKW}^+}(\mathcal{A})$, the adversary $\mathcal{A}^Q$ has access to a set of algorithms $Q = \{\mathsf{New}, \mathsf{Wrap}, \mathsf{Wrap\$}, \mathsf{Unwrap}, \mathsf{Punc}, \mathsf{Drv}, \mathsf{Corrupt}\}$, Wrap\$, which, given a key index i, a tag T, additional data AD, and a message m chosen by the adversary, responds with either a legitimate wrapping of m under the secret key sk_i or a random bit-string of length $|m|$. Similarly, the adversary can use real wrappings and key derivation, i.e. Wrap, Unwrap&Drv, without the need for puncturing through an additional oracle Wrap. However, in the case of key indistinguishability ($\mathbf{G}^{\mathbf{KIND}}_{\mathrm{PKW}^+}(\mathcal{A})$), the adversary has access to $Q = \{\mathsf{New}, \mathsf{Wrap}, \mathsf{Punc}, \mathsf{Drv}, \mathsf{Drv\$}\}$, where (Drv\$) is challenge key deriving oracle, which, given a key index i, a tag T and additional data AD, responds with either a legitimate wrapping of k under the secret key sk_i or a random bit-string of length $|k|$. The idea behind this approach is to ensure forward security for all messages wrapped using keys that have been punctured at the time of compromise and to account for potential information leakage from unpunctured ciphertexts that the adversary gains insight

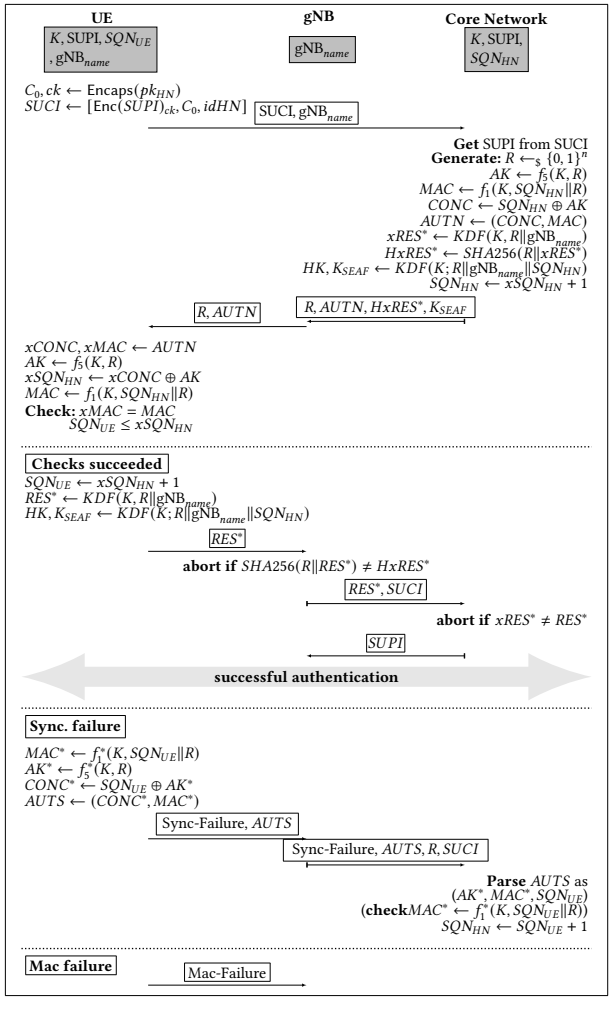


Figure 8: 5G-AKA Protocol

into during corruption. In other words, we aim to guarantee a degree of independence among key wrappings produced with distinct tags. The concept of forward security is implemented through a corruption oracle Corrupt, allowing the adversary to compromise the current version of a secret key sk_i, provided that all tags used in challenge queries under sk_i must be punctured at the time of corruption, as dictated by the puncturing oracle Punc. Furthermore, in this context, we explicitly treat ciphertexts under punctured tags as valid forgery attempts, even if they were previously produced by Wrap. This ensures that once a tag is punctured, no ciphertext with that tag will be accepted, providing a form of replay protection. The adversary wins if he is able to create a ciphertext (along with a tag and additional data) that was not generated by Wrap or for which the tag was punctured through Punc, and that, upon unwrapping, does not result in $\perp$.

Definition 12 (PKW^+ confidentiality and integrity). *Assuming* PKW^+ *is a puncturable key-wrapping scheme, we specify the*

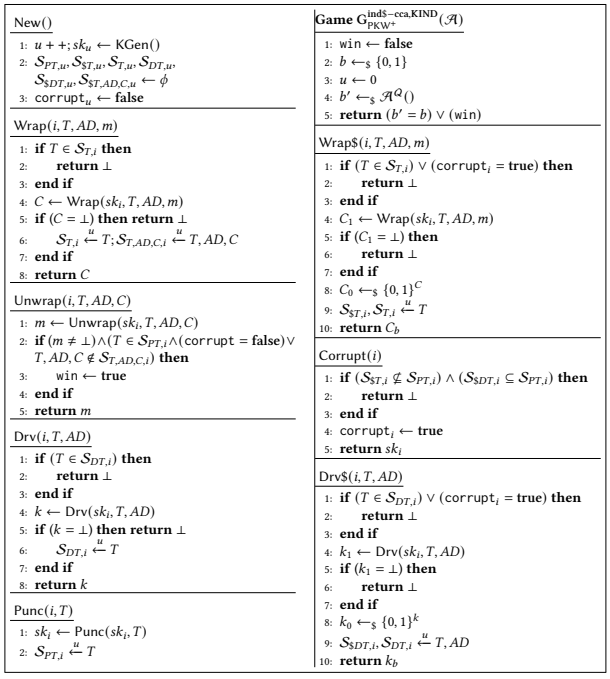


Figure 9: PKW+ security

advantage for a PPT algorithm $\mathcal{A}$ *has in terms of forward indistinguishability as:*

$$\mathbf{Adv}^{ind\$-cca,\mathsf{KIND}}_{\mathsf{PKW}^+} = 2|\mathbf{Pr}[\mathbf{G}^{\mathbf{ind\$-cca},\mathsf{KIND}}_{\mathsf{PKW}^+}(\mathcal{A}) \Rightarrow \mathbf{true}] - \frac{1}{2}| \quad (5)$$

, where $\mathbf{G}^{\mathbf{ind\$-cca},\mathsf{KIND}}_{\mathsf{PKW}^+}$ *is defined in* ***Figure*** *9. We say that* PKW^+ *is (ind\$ − cca,* KIND*) secure if, for all* $\mathcal{A}$*, the advantage is negligible in the security parameter* λ*.*

A.1 Instantiation of PKW+

Here, we present the construction framework for a PKW^+ scheme, illustrated in **Figure** 1. This framework utilizes an authenticated encryption scheme with associated data (AEAD) to ensure the secure encryption of messages. Each wrapping algorithm involves a secret key, additional data (AD), a tag (T), and the message to be wrapped. The AEADsecret key is generated by a key derivation function (KDF), which takes as input a key produced by a pseudorandom function (PPRF) and additional data (AD). The PPRF key is, in turn, generated using the secret key and tag (T). Additionally, the introduced derivation algorithm in PKW^+ (DRV()) utilizes the key generated by the PPRF, combined with the wrap tag, as input to the key derivation function (KDF), producing a new key for future use. This design approach facilitates the forward security of secret keys by puncturing the PPRF key, ensuring the unrecoverability of ciphertexts.

Next, we demonstrate that, under specific attributes of the foundational PPRF,AEADand KDFschemes, our design PKW^+[PPRF, KDF, AEAD] attains both puncture invariance and consistency (as per Theorem 5). Additionally, it achieves forward indistinguishability contingent upon the inherent strength of the PPRF and ensures the confidentiality of ciphertexts (Theorem 6).

Theorem 5. *(PKW$^+$ is consistent and puncture invariant) The new key-wrapping scheme* PKW$^+$[PPRF, KDF, AEAD] *illustrated in* ***Figure*** *1 is consistent, as specified in Definition 4. Furthermore, assuming the puncture invariance property of the PPRF, it follows that* PKW$^+$[PPRF, KDF, AEAD] *also holds the property of puncture invariance, in accordance with Definition 3.*

Proof: The puncture invariance of PKW$^+$[PPRF, KDF, AEAD] is a direct consequence of the puncture invariance inherent in PPRF. The consistency of PKW$^+$[PPRF, KDF, AEAD] is based on several crucial components: Firstly, the correctness of PPRFguarantees that its evaluation at a specific point remains unchanged even when other points are punctured, ensuring that the KDFkeys derived from a tag T remain unaffected by the puncturing of other tags. Secondly, the security of KDFguarantee the robustness of the keys generated for both AEADencryption and future processes, none of which are influenced by the puncturing of other tags. Lastly, the determinism of the encryption algorithm in the AEADscheme ensures that the ciphertext relies solely on the inputs to AEADencryption. Collectively, these factors conclusively establish the consistency and puncture invariance of PKW$^+$[PPRF, KDF, AEAD].

To accomplish this, we refer to the correctness definition of PPRF in [8], which asserts that for every subset of elements in the domain (Here, the tag space of PKW$^+$) denoted as $\{T_1, T_2, ..., T_n\} \subset \mathcal{T}$ and all $T \notin \setminus(T_1, ..., T_n)$, the following holds:

$$\mathbf{Pr}[\mathsf{Eval}(sk_0, T) = \mathsf{Eval}(sk_n, T) | sk_0 \leftarrow_{\$} \mathsf{PPRF.KGen}()] = 1 \quad (6)$$

where sk_n is derived by executing $sk_i \leftarrow_{\$} \mathsf{PPRF.Punc}(sk_{i-1}, T_i)$ for $i \in \{1, \ldots, n\}$. Now, to ensure consistency, the requirement precisely states that for all $AD \in \mathcal{AD}$ and all $m \in \mathcal{M}$,

$$\mathbf{Pr}[\mathsf{Wrap}(sk_0, T, AD, m) = \mathsf{Wrap}(sk_n, T, AD, m) | sk_0 \leftarrow_{\$} \mathsf{KGen}()] = 1 \quad (7)$$

Thanks to PPRFcorrectness this condition is satisfied for PKW$^+$[PPRF, KDF, AEAD], as per the definition of the construction, where:

(1) PKW$^+$.KGen() := PPRF.KGen()
(2) PKW$^+$.Punc(sk, T) := PPRF.Punc(sk, T)
(3) PKW$^+$.Drv(sk, T, AD) := KDF(PPRF.Eval(sk, T), AD)
(4) PKW$^+$.Wrap(sk_0, T, AD, m) := AEAD.Enc(KDF(PPRF.Eval(sk_0, T), AD), T, AD, m) = AEAD.Enc(KDF(PPRF.Eval(sk_n, T), AD), T, AD, m) := PKW$^+$.Wrap(sk_n, T, AD, m)

Theorem 6. *(PKW$^+$ is ind$\$ - cca$ secure) Consider the new key-wrapping scheme* PKW$^+$[PPRF, KDF, AEAD] *illustrated in* ***Figure*** *1. For any adversary $\mathcal{A}$ attempting to compromise the ind$\$ - cca$ security of* PKW$^+$[PPRF, KDF, AEAD]*, defined in 12, by querying oracles* New, Wrap$, Wrap, Unwrap, Punc, Corrupt *and* Drv *there exist corresponding adversaries ($\mathcal{B}_{pprf}, \mathcal{B}_{kdf}$, and$\mathcal{B}_{aead}$) that operate approximately at the same time as $\mathcal{A}$, such that:*

$$\mathbf{Adv}^{ind-cpa}_{\mathsf{PKW}^+}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{PPRF}}(\mathcal{B}_{pprf}) + \mathbf{Adv}_{\mathsf{KDF}}(\mathcal{B}_{kdf}) + \mathbf{Adv}_{\mathsf{AEAD}}(\mathcal{B}_{aead}) \quad (8)$$

Proof: To prove theorem 6, we divide it into two cases, denoted by $\mathbf{Adv}^{c_1}_{\mathsf{PKW}^+}(\lambda)$ and $\mathbf{Adv}^{c_2}_{\mathsf{PKW}^+}(\lambda)$. In c_1, we demonstrate that PKW$^+$ ensures the integrity of ciphertexts by proving that the probability of the adversary $\mathcal{A}$ winning in game G($ind\$ - cca$) is negligible. Similarly, in c_2, we establish that PKW$^+$ provides confidentiality, ensuring that ciphertexts are indistinguishable from random strings. As a result, the probability of the adversary $\mathcal{A}$ correctly guessing the bit b in game G($ind\$ - cca$) is also negligible. We then bound the advantage of $\mathcal{A}$ winning the game to: $\mathbf{Adv}^{ind\$-cca}_{\mathsf{PKW}^+}(\lambda) \leq \mathbf{Adv}^{C_1}_{\mathsf{PKW}^+}(\lambda) + \mathbf{Adv}^{C_2}_{\mathsf{PKW}^+}(\lambda)$.

Case 1: Let game G_0 be equivalent to G($ind\$ - cca$), with the algorithms of PKW$^+$[PPRF; KDF; AEAD] implemented directly using the underlying PPRF, KDF and AEAD schemes. We begin by exploiting the forward pseudorandomness security property of PPRF and substitute the output of PPRF keys with random keys. Subsequently, in the following Game, we utilize the randomized PPRF keys generated previously to derive keys using PKW$^+$ .Drv(), arguing that the resulting keys are indistinguishable from random keys. Next, we exploit the unforgeability/ integrity security of AEAD ciphertexts and substitute the output of ciphertext with random, and we argue that the $\mathcal{A}$ has a negligible advantage of forging a valid ciphertext. The reduction works as follows:

Game 0: This the original game $ind\$ - cca$ described in 9:

$$\mathbf{Adv}^{ind\$cca}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}$$

Game 1: In this game, we introduce an abort event that triggers if $\mathcal{A}$ sends Unwrap(j, ..) query under index j, where $j \in \{0, \ldots, n\}$ and setting `win` to **true**. We begin this game by guessing the key index (i) and ($i \neq j$) of the first successful forgery (i.e. `win` = **true**). Thus, $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n \cdot \mathbf{Adv}_{G_1}.$$

Game 2: In this security game, we exploit the forward pseudorandomness (FPR) security property of PPRF by replacing the output of PPRF key (i.e. sk_a in PKW$^+$ construction) with a random key. To accomplish this, we introduce a reduction $\mathcal{B}_1$ which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a (PPRF) challenger C_{PPRF}. $\mathcal{B}_1$ simulates ($G^{fpr-ro\$}_{\mathsf{PPRF}}$). For all key indices $j \neq i$, C_{PPRF} generates sk_j and responds to queries using real oracles. For key index i $\mathcal{B}_1$ uses $Ro\$ - \mathsf{Eval}$ to generate keys for Wrap, Unwrap and Drv queries. Specifically: Upon receiving wrap/unwrap queries Wrap/Unwrap($i, T, AD, m/c$) from $\mathcal{A}$, $\mathcal{B}_1$ queries C_{PPRF} to obtain a real or random k_i using $Ro\$ - \mathsf{Eval}(i, T)$. Subsequently, $\mathcal{B}_1$ utilises the output key from C_{PPRF} to generate another key using KDF. This generated key is then employed to encrypt/decrypt the message/ciphertext using AEAD algorithms. Finally, $\mathcal{B}_1$ replies to the $\mathcal{A}$ with the resulting message/ciphertext. However, when $\mathcal{A}$ calls derive query (Drv), we obtain the random key similarly and use the KDF to output a distinct key that has been used in the AEAD. Note that, the $\mathcal{A}$ cannot distinguish between multiple calls to $RO\$ - \mathsf{Eval}$ as it is handled internally in the PPRF game. The $\mathcal{A}$ simulation in Game 1 is essentially the same as Game 2, with the exception of replacing PPRF keys with random keys. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 1** and **Game 2**, then $\mathcal{A}$ has effectively breaks the fpr security of PPRF. Thus:

$$\mathbf{Adv}_{G_1} \leq \mathbf{Adv}_{G_2} + \mathbf{Adv}^{fpr}_{\mathcal{B}_1,\mathsf{PPRF}}$$

Game 3: In this game, we replace the output of the KDF (i.e. sk_b, sk_c in PKW$^+$ construction) with uniformly random value. We

do so by interacting with a $\mathcal{B}_2$ that interacts with C_{KDF} challenger. Similar to Game 2, for key index i $\mathcal{B}_2$ uses the output of KDF to generate keys for Wrap, Unwrap and Drv queries. Whenever sk_a is used in Wrap, Unwrap or Drv to generate sk_b, sk_c, $\mathcal{B}_2$ first we check if Wrap, Unwrap and Drv were called since the last puncture ($\mathsf{Punc}(i, ..)$) query. If this condition holds, $\mathcal{B}_2$ then calls C_{KDF} to obtain random keys $s\hat{k}_b, s\hat{k}_c$ and add the output to the lookup table. Otherwise we do a lookup for $\mathrm{T}[sk_a, AD]$ if it returns $\perp$, then $\mathcal{B}_2$ calls C_{KDF} to obtain random keys $s\hat{k}_b, s\hat{k}_c$ and add the output to the lookup table. Alternatively, $\mathcal{B}_2$ uses *out* as output values. since by **Game 2** the sk_a is already uniformly random, this change is sound. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 2** and **Game 3**, then $\mathcal{A}$ has effectively breaks the security of KDF, thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}_{\mathcal{B}_2,\mathsf{KDF}}$$

Game 4: Here, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C that decrypts/unwraps correctly and was not generated by Wrap, breaking the unforgability/ integrity security of AEAD ciphertexts. We do so by defining a reduction $\mathcal{B}_3$ that initialises an AEAD challenger C_{AEAD}. Upon receiving wrap/unwrap queries $\mathsf{Wrap/Unwrap}(sk, T, AD, m/c)$ from $\mathcal{A}$, $\mathcal{B}_3$ first check if Wrap, Unwrap were called since the last puncture ($\mathsf{Punc}(i, ..)$) query, if not we retrieve the previously computed output. Otherwise, $\mathcal{B}_3$ queries C_{AEAD} to obtain a real or random ciphertext C using $Ro\$(sk_b, n, AD, m)$. Finally, $\mathcal{B}_3$ replies to the $\mathcal{A}$ with the resulting ciphertext. This way $\mathcal{A}$ wins if it submits a valid forgery (C) to Unwrap, where correctly output m (i.e., $\mathsf{Unwrap}(i, T, AD, C) \rightarrow m : m \neq \perp$), and if either C has never been output from Wrap (i.e $T, AD, C \notin \mathcal{S}_{T,AD,C,i}$) or if tag T was previously punctured via Punc (i.e., $T \in \mathcal{S}_{PT,i}$). since by **Game 3** the sk_b is already uniformly random, this change is sound. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 3** and **Game 4**, then $\mathcal{A}$ has effectively breaks the security of AEAD. This implies: $\mathbf{Adv}_{G_3} \leq \mathbf{Adv}^{unf}_{G_4} + \mathbf{Adv}_{\mathcal{B}_3,\mathsf{AEAD}}$

Since as per Case 1 the $\mathcal{A}$ has a negligible advantage of breaking AEAD security, where we abort if the $\mathcal{A}$ can forge a valid ciphertext, Thus

$$\mathbf{Adv}^{c_1}_{\mathsf{PKW}^+}(\mathcal{A}) = 0$$

We now bound the advantage of $\mathcal{A}$ in **Case 2**.

Case 2: Let game G_0 be equivalent to $\mathrm{G}(ind\$ - cca)$, with the algorithms of PKW^+ [PPRF; KDF; AEAD] implemented directly using the underlying PPRF, KDFand AEAD schemes. we begin by exploiting the forward pseudorandomness security property of PPRF and substitute the output of PPRF keys with random keys. Subsequently, in the following Game, we utilize the randomized PPRF keys generated previously to derive keys using $\mathsf{PKW}^+.\mathsf{Drv}()$, arguing that the resulting keys are indistinguishable from random keys. Following this, we employ the derived KDF keys to encrypt messages using $\mathsf{PKW}^+.\mathsf{Wrap}\$()$, and we argue that the resulting ciphertext is indistinguishable from random. The reduction works as follows:

Game 0: This the original game $ind\$ - cca$ described in 9:

$$\mathbf{Adv}^{ind\$]cca}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}$$

Game 1: In this game, we introduce an abort event that triggers if $\mathcal{A}$ sends $\mathsf{Unwrap}(j, ..)$ query under index j, where $j \in \{0, \ldots, n\}$ and setting `win` to **true**. We begin this game by guessing the key index (i) and ($i \neq j$) of the first successful forgery (i.e. `win` = **true**). Thus, $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n \cdot \mathbf{Adv}_{G_1}.$$

Game 2: In this security game, we exploit the forward pseudorandomness (FPR) security property of PPRF by replacing the output of PPRF key (i.e. sk_a in PKW^+ construction) with a random key. To accomplish this, we introduce a reduction $\mathcal{B}_1$ which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a (PPRF) challenger C_{PPRF}. $\mathcal{B}_1$ simulates ($G^{fpr-ro\$}_{\mathsf{PPRF}}$). For all key indices $j \neq i$, C_{PPRF} generates sk_j and responds to queries using real oracles. For key index i $\mathcal{B}_1$ uses $Ro\$ - \mathsf{Eval}$ to generate keys for Wrap, Unwrap and Drv queries. Specifically: Upon receiving wrap/unwrap queries $\mathsf{Wrap/Unwrap}(i, T, AD, m/c)$ from $\mathcal{A}$, $\mathcal{B}_1$ queries C_{PPRF} to obtain a real or random k_i using $Ro\$ - \mathsf{Eval}(i, T)$. Subsequently, $\mathcal{B}_1$ utilises the output key from C_{PPRF} to generate another key using KDF. This generated key is then employed to encrypt/decrypt the message/ciphertext using AEAD algorithms. Finally, $\mathcal{B}_1$ replies to the $\mathcal{A}$ with the resulting message/ciphertext. However, when $\mathcal{A}$ calls derive query (Drv), we obtain the random key similarly and use the KDF to output a distinct key that has been used in the AEAD. Note that, the $\mathcal{A}$ cannot distinguish between multiple calls to $RO\$ - \mathsf{Eval}$ as it is handled internally in the PPRF game. The $\mathcal{A}$ simulation in Game 1 is essentially the same as Game 2, with the exception of replacing PPRF keys with random keys. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 1** and **Game 2**, then $\mathcal{A}$ has effectively breaks the fpr security of PPRF. Thus:

$$\mathbf{Adv}_{G_1} \leq \mathbf{Adv}_{G_2} + \mathbf{Adv}^{fpr}_{\mathcal{B}_1,\mathsf{PPRF}}$$

Game 3: In this game, we replace the output of the KDF (i.e. sk_b, sk_c in PKW^+ construction) with uniformly random value. We do so by interacting with a $\mathcal{B}_2$ that interacts with C_{KDF} challenger. Similar to Game 2, for key index i $\mathcal{B}_2$ uses the output of KDF to generate keys for Wrap, Unwrap and Drv queries. Whenever sk_a is used in Wrap, Unwrap or Drv to generate sk_b, sk_c, $\mathcal{B}_2$ first we check if Wrap, Unwrap and Drv were called since the last puncture ($\mathsf{Punc}(i, ..)$) query. If this condition holds, $\mathcal{B}_2$ then calls C_{KDF} to obtain random keys $s\hat{k}_b, s\hat{k}_c$ and add the output to the lookup table. Otherwise we do a lookup for $\mathrm{T}[sk_a, AD]$ if it returns $\perp$, then $\mathcal{B}_2$ calls C_{KDF} to obtain random keys $s\hat{k}_b, s\hat{k}_c$ and add the output to the lookup table. Alternatively, $\mathcal{B}_2$ uses *out* as output values. since by **Game 2** the sk_a is already uniformly random, this change is sound. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 2** and **Game 3**, then $\mathcal{A}$ has effectively breaks the security of KDF, thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}_{\mathcal{B}_2,\mathsf{KDF}}$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ distinguishes a real ciphertext from a random ciphertext C, breaking the confidentiality security of AEAD. We do so by defining a reduction $\mathcal{B}_3$ that initialises an AEAD challenger C_{AEAD}. Upon receiving wrap/unwrap queries $\mathsf{Wrap/Unwrap}(sk, T, AD, m/c)$ from $\mathcal{A}$, $\mathcal{B}_3$ first check if Wrap, Unwrap were called since the last puncture ($\mathsf{Punc}(i, ..)$) query, if not we retrieve the previously computed output. Otherwise, $\mathcal{B}_3$ queries C_{AEAD} to obtain a real or random ciphertext C using $Ro\$(sk_b, n, AD, m)$. Finally, $\mathcal{B}_3$ replies to the $\mathcal{A}$ with the resulting ciphertext. This way $\mathcal{A}$ wins if he manages to

distinguish the result from real or random. since by **Game 3** the sk_b is already uniformly random, this change is sound. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 3** and **Game 4**, then $\mathcal{A}$ has effectively breaks the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}^{unf}_{G_4} + \mathbf{Adv}_{\mathcal{B}_3,\mathsf{AEAD}}$$

Here we emphasise that as a result of these changes, the generated ciphertext is indistinguishable from a random ciphertext, thus $\mathcal{A}$ has a negligible advantage in winning the game:

$$\mathbf{Adv}^{c_2}_{\mathsf{PKW}^+}(\mathcal{A}) = 0$$

Theorem 7. *(*PKW^+ *is* KIND *secure) Consider the new key-wrapping scheme* $\mathsf{PKW}^+[\mathsf{PPRF}, \mathsf{KDF}, \mathsf{AEAD}]$ *illustrated in* ***Figure 1****. For any adversary* $\mathcal{A}$ *attempting to compromise the* KIND *security of* $\mathsf{PKW}^+[\mathsf{PPRF}, \mathsf{KDF}, \mathsf{AEAD}]$*, defined in 12, by querying oracles* New, Wrap, Punc, Drv$ *and* Drv*, there exist corresponding adversaries (*$\mathcal{B}_{pprf}$ *and* $\mathcal{B}_{kdf}$*) that operate approximately at the same time as* $\mathcal{A}$*, such that:*

$$\mathbf{Adv}^{\mathsf{KIND}}_{\mathsf{PKW}^+}(\mathcal{A}) \quad \leq \quad \mathbf{Adv}_{\mathsf{PPRF}}(\mathcal{B}_{pprf}) \; + \; \mathbf{Adv}_{\mathsf{KDF}}(\mathcal{B}_{kdf}) \quad (9)$$

Proof: To prove theorem 7, we demonstrate that PKW^+ ensures key indistinguishability by proving that the probability of the adversary $\mathcal{A}$ winning in game G(KIND) is negligible.

Game 0: This the original game KIND described in 9:

$$\mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}$$

Game 1: In this game, we introduce an abort event that triggers if $\mathcal{A}$ sends $\mathsf{Unwrap}(j, ..)$ query under index j, where $j \in \{0, \ldots, n\}$ and setting `win` to **true**. We begin this game by guessing the key index (i) and ($i \neq j$) of the first successful forgery (i.e. `win` = **true**). Thus, $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n \cdot \mathbf{Adv}_{G_1}.$$

Game 2: In this security game, we exploit the forward pseudorandomness (FPR) security property of PPRF by replacing the output of PPRF key (i.e. sk_a in PKW^+ construction) with a random key. To accomplish this, we introduce a reduction $\mathcal{B}_1$ which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a (PPRF) challenger C_{PPRF}. $\mathcal{B}_1$ simulates ($G^{fpr-ro\$}_{\mathsf{PPRF}}$). For all key indices $j \neq i$, C_{PPRF} generates sk_j and responds to queries using real oracles. For key index i $\mathcal{B}_1$ uses $Ro\$ - \mathsf{Eval}$ to generate keys for Wrap, Unwrap and Drv queries. Specifically: Upon receiving wrap/unwrap queries $\mathsf{Wrap}/\mathsf{Unwrap}(i, T, AD, m/c)$ from $\mathcal{A}$, $\mathcal{B}_1$ queries C_{PPRF} to obtain a real or random k_i using $Ro\$ - \mathsf{Eval}(i, T)$. Subsequently, $\mathcal{B}_1$ utilises the output key from C_{PPRF} to generate another key using KDF. This generated key is then employed to encrypt/decrypt the message/ciphertext using AEAD algorithms. Finally, $\mathcal{B}_1$ replies to the $\mathcal{A}$ with the resulting message/ciphertext. However, when $\mathcal{A}$ calls derive query (Drv), we obtain the random key similarly and use the KDF to output a distinct key that has been used in the AEAD. Note that, the $\mathcal{A}$ cannot distinguish between multiple calls to $RO\$ - \mathsf{Eval}$ as it is handled internally in the PPRF game. The $\mathcal{A}$ simulation in Game 1 is essentially the same as Game 2, with the exception of replacing PPRF keys with random keys. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 1** and **Game 2**, then $\mathcal{A}$ has effectively breaks the fpr security of PPRF. Thus:

$$\mathbf{Adv}_{G_1} \leq \mathbf{Adv}_{G_2} + \mathbf{Adv}^{fpr}_{\mathcal{B}_1,\mathsf{PPRF}}$$

Game 3: In this game, we replace the output of the KDF (i.e. sk_b, sk_c in PKW^+ construction) with uniformly random value. We do so by interacting with a $\mathcal{B}_2$ that interacts with C_{KDF} challenger. Similar to Game 2, for key index i $\mathcal{B}_2$ uses the output of KDF to generate keys for Wrap, Unwrap and Drv queries. Whenever sk_a is used in Wrap, Unwrap or Drv to generate sk_b, sk_c, $\mathcal{B}_2$ first we check if Wrap, Unwrap and Drv were called since the last puncture ($\mathsf{Punc}(i, ..)$) query. If this condition holds, $\mathcal{B}_2$ then calls C_{KDF} to obtain random keys $\hat{sk_b}, \hat{sk_c}$ and add the output to the lookup table. Otherwise we do a lookup for $\mathrm{T}[sk_a, AD]$ if it returns $\perp$, then $\mathcal{B}_2$ calls C_{KDF} to obtain random keys $\hat{sk_b}, \hat{sk_c}$ and add the output to the lookup table. Alternatively, $\mathcal{B}_2$ uses *out* as output values. since by **Game 2** the sk_a is already uniformly random, this change is sound. Therefore, if $\mathcal{A}$ manages to distinguish between **Game 2** and **Game 3**, then $\mathcal{A}$ has effectively breaks the security of KDF, thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}_{\mathcal{B}_2,\mathsf{KDF}}$$

Here we emphasise that as a result of these changes, the generated key in **Game 3** is now uniformly random and independent regardless of the bit b sampled by C, thus $\mathcal{A}$ has no advantage in guessing the bit b:

$$\mathbf{Adv}^{\mathsf{KIND}}_{\mathsf{PKW}^+}(\mathcal{A}) = 0.$$

A.2 Puncturable PRFs

Puncturable Pseudo-Random Functions (PPRFs) have been proposed and defined in various research papers, including references [8, 10, 23]. Our focus in this context is on PPRFs with deterministic puncturing algorithms.

Definition 13 (PPRF). *A puncturable pseudorandom function (*PPRF*) consists of three algorithms -* KGen, Eval, *and* Punc- *paired with three associated sets: the secret-key space (*$\mathcal{SK}$*), the domain (*$\mathcal{X}$*), and the range (*$\mathcal{Y}$*).* ‡

- $sk \leftarrow_\$ \mathsf{KGen}()$*, a probabilistic key generation algorithm, takes no input and outputs the secret key* $sk \in \mathcal{SK}$*.*
- $y/\perp \leftarrow \mathsf{Eval}(sk; x)$*, the function evaluation algorithm Eval, on input the secret key sk and an element* $x \in \mathcal{X}$ *outputs* $y \in \mathcal{Y}$ *or, to indicate failure,* $\perp$
- $sk' \leftarrow \mathsf{Punc}(sk; x)$*, a deterministic puncturing algorithm* Punc*, on input the secret keysk and an element* $x \in \mathcal{X}$ *outputs an updated secret key* $sk \in \mathcal{SK}$*.*

For PPRF correctness, it is require that for all $sk \in \mathcal{SK}$ *and every* §, $y \in \mathcal{Y}$*:*

(1) $\mathbf{Pr}[\mathsf{Eval}(sk_0, x)] \neq \perp \| sk_0 \leftarrow_\$ \mathsf{KGen}()] = 1$

(2) $If sk' \leftarrow \mathsf{Punc}(sk; x) and y \neq x, then \mathsf{Eval}(sk; y) = \mathsf{Eval}(sk'; y).$

(3) $If sk' \leftarrow \mathsf{Punc}(sk; x), then Eval(sk'; x) = \perp$

A.3 Key Derivation Function

A key derivation function ($\mathsf{KDF}(r, L, s, c) \rightarrow k^{|L|}$) is a deterministic algorithm, taking four inputs: a random seed r, a length L, salt s and context c (where s and c are optional), and returning a key k of L bits. We capture the KDF security of KDF [17] via the following game played between a challenger C, and PPT $\mathcal{A}$:

‡For the security proof of the PPRF, please refer to [23] or [4].

(1) C generates (r, α), where r is a random value sampled from distribution α. Next, C generates a uniformly random salt s, and returns (α, s) to $\mathcal{A}$.
(2) $\mathcal{A}$ queries arbitrary (c_i, l_i) to C, who returns $k_i = \mathsf{KDF}(r, l_i, s, c_i)$.
(3) Next, $\mathcal{A}$ chooses l and $c \notin (c_1.....c_t)$. C samples a bit $b \leftarrow_\$ \{0,1\}$, and computes $k_0 = \mathsf{KDF}(r, l, s, c)$ and $k_1 \leftarrow_\$ \{0,1\}^l$. C returns k_b to $\mathcal{A}$.
(4) $\mathcal{A}$ queries arbitrary (c_i, l_i) (where $c \notin c_i$) to C, who returns $k_i = \mathsf{KDF}(r, l_i, s, c_i)$.
(5) $\mathcal{A}$ outputs a guess bit b'.

We say that KDF is KDF secure if the advantage $\mathbf{Adv}^{\mathrm{KDF}}_{\mathrm{KDF}}(\mathcal{A})$ of $\mathcal{A}$ in winning the KDF game is negligible, where:

$$\mathbf{Adv}^{\mathrm{KDF}}_{\mathrm{KDF}}(\mathcal{A}) = 2 \cdot |\mathbf{Pr}[(b = b')] - \frac{1}{2}|$$

A.4 Authenticated Encryption with Associated Data

This work uses authenticated encryption with associated data (AEAD), ensuring the integrity and authenticity of non-encrypted data. Specifically, we follow Rogaway's nonce-based AEAD scheme [21].

Definition 14. *[Authenticated Encryption with Associated Data] The AEAD consists of a pair of two algorithms AEAD* : {Enc, Dec} *associated with four sets: the secret-key space $\mathcal{SK}$, the nonce space $\mathcal{N}$, the associated data space $\mathcal{AD}$ and the message space $\mathcal{M}$.*

- $\mathsf{Enc}(sk, n, ad, m) \rightarrow c$*: a deterministic encryption algorithm takes an input of a secret key $sk \in \mathcal{SK}$, a nonce $n \in \mathcal{N}$, an associated data $ad \in \mathcal{AD}$, and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \in \{0,1\}^{|m|}$.*
- $\mathsf{Dec}(sk, n, ad, c) \rightarrow m/\perp$*: a deterministic decryption algorithm takes an input of a secret key $sk \in \mathcal{SK}$, a nonce $n \in \mathcal{N}$, an associated data $ad \in \mathcal{AD}$, and a ciphertext $c \in \{0,1\}^*$, and outputs either a message $m \in \mathcal{M}$ or $\perp$ for failure.*

B Full Security analysis of PGUP

B.1 Mutual Authentication

B.1.1 ***MA-security of PGUP AKA (Proof of Theorem1).*** We begin by showing that the PGUP AKA protocol achieves mutual authentication. **Proof**: Our proof is divided into three cases, denoted by $\mathbf{Adv}^{\mathrm{MA,clean},c_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$, $\mathbf{Adv}^{\mathrm{MA,clean},c_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ and $\mathbf{Adv}^{\mathrm{MA,clean},c_3}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$

We then bound the advantage of $\mathcal{A}$ winning the game under certain assumptions to $\mathbf{Adv}^{\mathrm{MA,clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq (\mathbf{Adv}^{\mathrm{MA,clean},c_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathrm{MA,clean},c_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathrm{MA,clean},c_3}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda))$.

- **Case 1:** assume that the first **clean** session π^s_i to accept without a matching session (i.e. $\pi^s_i.\rho$ = gNB).
- **Case 2:** assume that the first **clean** session π^s_i to accept without a matching session(i.e. $\pi^s_i.\rho$ = UE).
- **Case 3:** assume that the first **clean** session π^s_i to accept without a matching session(i.e. $\pi^s_i.\rho$ = CN).

Case 1: According to the definition of this case, gNB either accepts messages $\mathbf{m_0}$, $\mathbf{m_4}$ or $\mathbf{m_7}$, without a matching subset (i.e. without honest UE partner). Note that in this case, $\mathcal{A}$ cannot corrupt the long-term UE symmetric-key.

Game 0: This is the original mutual authentication game described in 7.3:

$$\mathbf{Adv}^{\mathrm{MA,clean,C1}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(i, s) \in n_P \times n_S$ of the first gNB session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index t of the CN session π^{CN}_t and we abort if there exists $\pi^{\mathrm{CN}}_{t^*}$ that matches the π^s_i and $(t \neq t^*)$, introducing a factor of n_S in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_1} \leq n_S \mathbf{Adv}_{G_2}.$$

Game 3 : In this game, we guess UE party index j such that $\pi^{\mathrm{CN}}_t.pid = j$, introducing a factor of n_P in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_2} \leq n_P \mathbf{Adv}_{G_3}.$$

Game 4 : In this game, we guess session index $r \in n_S$ and aborts if π^j_r registers a ciphertext in $C_{txt} \leftarrow (C, T, AD, m, j)$ and π^{CN}_t computes $rk \leftarrow \mathsf{Drv}(k_i, T', C_0)$ and $(., T', ., .) \in C_{txt}$ but $T' \neq T$, introducing a factor of n_S in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_3} \leq n_S \mathbf{Adv}_{G_4}.$$

Game 5 : In this game, we replace the computation of rk in π^j_r and π^{CN}_t sessions with uniformly random value ($\hat{rk}$). We do so by introducing a reduction $\mathcal{B}_1$, which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a PKW^+ challenger C_{PKW^+}. Every time $\mathcal{A}$ calls New(), $\mathcal{B}_1$ calls to PKW^+.New(), then we assign i to each $SUTI$ and a table $(SUTI, i)$ is maintained.

Additionally, $\mathcal{B}_1$ maintains the following registers C_{txt}, R_k and T_p. C_{txt} will be updated whenever we call $\mathsf{Wrap}(T, AD, m, i) \rightarrow C$ on a new entry, such that $(\cdot, T, AD, m, i) \notin C_{txt}$. Similarly, the R_k register will be updated whenever we call $\mathsf{Drv}(T, AD, i) \rightarrow rk$ on new entries, such that $(\cdot, T, AD, i) \notin R_k$. Finally, T_p will be updated whenever we call $\mathsf{Punc}(T, i, \rho)$.

Whenever $\mathbf{Send}(i, s, m, \rho)$ is called, we need first to check the T_p register and follow these conditions:

- IF $(T, i, \rho) \in T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0 \| C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0 \| C_1, SUTI||\Delta)$ by $\mathcal{B}_1$. and if Unwrap was queried, the computation of $m \leftarrow \mathsf{Unwrap}(K, T, C_0 \| C_1, C_2)$ is replaced with a call to Unwrap $(i, T, C_0 \| C_1, C_2)$ by $\mathcal{B}_1$. Similarly, whenever $\mathsf{Drv}(K, T, C_0 \| C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0 \| C_1)$.
- IF $(T, i, \overline{\rho} \in T_p$:Depending on the algorithm that was called, we check its registry for an output. If Unwrap or Drv were queried, we recover the output from the register (C_{txt}, R_k), respectively. However, if Wrap was queried, then the adversary has either issued a Corrupt, hence $\mathcal{B}_1$ can simulate Wrap, or the adversary has forged a message between UE and CN, hence breaking the PKW^+ security.
- IF $(T, i, \cdot) \notin T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0 \| C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0 \| C_1, SUTI||\Delta)$ by $\mathcal{B}_1$. Then, the output is added to the register as $(C, i, T, C_0 \| C_1, SUTI||\Delta) \rightarrow C_{txt}$. Similarly, whenever $\mathsf{Drv}(K, T, C_0 \| C_1) \rightarrow$

rk is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0\|C_1)$. and adds it to the register as $\mathsf{Drv}(i, T, C_0\|C_1) \rightarrow R_K$.

We note that this process is precisely how all parties interact with their collective PKW$^+$ state, making this replacement sound. If $\mathcal{A}$ can distinguish between these two games, then $\mathcal{A}$ breaks the PKW$^+$ key indistinguishability game. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{B}_1,\mathsf{PKW}^+}.$$

Game 6: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_2 that decrypts correctly and is accepted by π_t^{CN}, but was not generated by π_r^j, breaking the authentication security of AEAD in PKW$^+$. In this game, no replacement is required because if the $\mathcal{A}$ could forge a cipher text, then they would win **Game 5**. Note that $(C_0\|C_1)$ is securely authenticated by PKW$^+$.Wrap() as additional data, and by the definition of **Case 1** $\mathcal{A}$ cannot issue **CorruptLTK**(UE). Any $\mathcal{A}$ that can trigger this abort event can be used to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_5} \leq \mathbf{Adv}_{G_6} + \mathbf{Adv}^{auth-aead}_{\mathsf{PKW}^+}.$$

Game 7: In this game we replace the authentication and mac keys AK, MK with uniformly random values $\hat{AK}, \hat{MK}$. We do so by defining a reduction a $\mathcal{B}_2$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{rk}$ and replacing the computation of AK, MK in any session that computes AK, MK, with the outputs from the C_{KDF} $\hat{AK}, \hat{MK}$. Since $AK, MK \leftarrow \mathsf{KDF}(rk, R)$ and by **Game 5** $\hat{rk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of KDF. Thus:

$$\mathbf{Adv}_{G_6} \leq \mathbf{Adv}_{G_7} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_2,\mathsf{KDF}}.$$

Game 8: In this game, we introduce an abort event that triggers if the same hash value exists for different inputs, thereby introducing hash collisions during the challenger execution with an honest session. We do so by defining a reduction $\mathcal{B}_3$ that initialises an Hash challenger C_{Hash} and maintains a lookup table. Whenever $\mathcal{B}_3$ needs to hash a value, he queries the lookup table on x and recovers (out). The abort event only triggers if $\mathcal{A}$ can get the same output hash values from two distinct input values, thus finding a collision and breaking the security of the Hash scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of Hash. This implies:

$$\mathbf{Adv}_{G_7} \leq \mathbf{Adv}_{G_8} + \mathbf{Adv}^{collision}_{\mathcal{B}_3,\mathsf{Hash}}.$$

Game 9: In this game, we replace the response and session keys RK, SK with uniformly random values $\hat{RK}, \hat{SK}$. We do so by defining a reduction a $\mathcal{B}_4$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{AK}$ and replacing the computation of RK, SK in any session that computes RK, SK, with the outputs from the C_{KDF} $\hat{RK}, \hat{SK}$. Since $RK, SK \leftarrow \mathsf{KDF}(\hat{AK}, SUTI)$ and by **Game 7**,$\hat{AK}$ is already uniformly random, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 8** from **Game 9** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_8} \leq \mathbf{Adv}_{G_9} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Game 10: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a valid tag (MAC), but (MAC) was not output by π_j^t. We do so by defining a reduction $\mathcal{B}_5$ that initialises an MAC challenger C_{MAC}, which $\mathcal{B}_5$ queries when π_j^t needs to MAC (SQN, R) with $(\hat{MK})$ and from **Game 7** $\hat{MK}$ is already uniformly random and independent. This abort event occurs if MAC is verified correctly under $\hat{MK}$, but MAC was never produced by the C_{MAC}, breaking the security of the MAC scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the existential unforgeability security of the MAC scheme. This implies:

$$\mathbf{Adv}_{G_9} \leq \mathbf{Adv}_{G_{19}} + \mathbf{Adv}^{euf-cma}_{\mathcal{B}_5,\mathsf{MAC}}.$$

Game 11: In this game, we replace the response (RES^*) and (K_{SEAF}, HK) with uniformly random value $(\hat{RES^*})$ and $(\hat{K_{SEAF}}\hat{HK})$, respectively. We do so by defining a reduction $\mathcal{B}_6$ that interacts with a KDF challenger C_{KDF}, querying C_{KDF} with $\hat{SK}, \hat{RK}$ and replacing the computation of (RES^*) and (K_{SEAF}, HK) in any session that computes RES^*, K_{SEAF}, HK, with the outputs from the $(C_{\mathsf{KDF}}$ $\hat{RK})$ and $(C_{\mathsf{KDF}}$ $\hat{K_{SEAF}}\hat{HK})$. Since $(RES^* \leftarrow \mathsf{KDF}(\hat{RK}, R\|\mathrm{gNB}_{name}))$ and $(C_{\mathsf{KDF}}$ $\hat{K_{SEAF}}\hat{HK})$ and by **Game 9** $\hat{RK}$ and $\hat{SK}$ are already uniformly random, so these changes are sound. Any $\mathcal{A}$ that can distinguish **Game 10** from **Game 11** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_{10}} \leq \mathbf{Adv}_{G_{11}} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_6,\mathsf{KDF}}.$$

By now, it becomes evident that the $\mathcal{A}$ is unable to tamper with messages exchanged between the π_i^s and π_j^t. Specifically, π_i^s exclusively accepts message $\mathbf{m_0}$ from an honest matching partner, and $\mathcal{A}$ cannot modify the content of C_2 without compromising the security of PKW$^+$. Additionally, $C_0\|C_1$ remains unaltered since it is authenticated using C_2. Especially since the $\mathcal{A}$ cannot corrupt the long-term UE symmetric-key, according to the definition of this case. Similarly, the adversary cannot modify $\mathbf{m_4}, \mathbf{m_7}$ without compromising the security of KDF&MAC. Thus, summing the probabilities we find that the $\mathcal{A}$ has negligible advantage in winning the MA-security experiment:

$$\mathbf{Adv}_{G_{11}} = 0$$

We now bound the advantage of $\mathcal{A}$ in **Case 2**. In this case, UE accepts messages $\mathbf{m_3}$ without a matching subset (no honest CN partner).

Case 2.1: According to the definition of this case, UE accepts message $\mathbf{m_3}$ without a matching subset (i.e. without honest CN partner). In this subcase, $\mathcal{A}$ cannot corrupt the long-term asymmetric secret key of CN.

Game 0: This is the original mutual authentication game described in 7.3:

$$\mathbf{Adv}^{\mathsf{MA},\mathbf{clean},\mathrm{C2}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1: In this game, we guess the index $(i, s) \in n_P \times n_S$ of the first UE session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \mathbf{Adv}_{G_1}.$$

Game 2: In this game, we guess the index $(j) \in n_S$ of the first CN session that accepts without a matching subset, introducing a factor of n_S in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_1} \leq n_S \mathbf{Adv}_{G_2}.$$

Game 3: In this game, we replace the key ck with uniformly random and independent value $\hat{ck}$.We do so by interacting with a

$\mathcal{B}_1$ challenger $C_{\mathsf{KEM}}^{ind-cpa}$, and replacing pk_{CN} with a public key pk recieved from $\mathcal{B}_1$. By the definition of the execution environment, pk_{CN} is generated uniformly random and independent, and by the definition of **Case 2.1** $\mathcal{A}$ cannot issue **CorruptASK**(CN) and this replacement is sound. Any $\mathcal{A}$ that can detect the replacement can be used to break $ind-cpa$ security of KEM. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{ind-cpa}_{\mathcal{B}_1,\mathsf{KEM}}.$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext R' (keyed by $\hat{ck}$) that decrypts correctly and was not generated by CN, breaking the authentication security of AEAD. We do so by defining a reduction $\mathcal{B}_2$ that initialises an AEAD challenger C_{AEAD}, which $\mathcal{B}_2$ queries when he needs to encrypt/decrypt with $\hat{ck}$. By **Game 3** $\hat{ck}$ is already uniformly random and independent, and this replacement is undetectable. Note that $AUTN$ is securely authenticated by AEAD security since it is included in the AEAD functions as additional data. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_2$ to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{auth-aead}_{\mathcal{B}_2,\mathsf{AEAD}}.$$

In this stage, it becomes evident that the $\mathcal{A}$ is unable to tamper with messages exchanged between the π_i^s and π_j^t. Specifically, π_i^s exclusively accepts message $\mathbf{m_3}$ from an honest matching partner, and $\mathcal{A}$ cannot modify the content of m_3 without compromising the security of AEAD&KEM. Additionally, $AUTN$ remains unaltered since it is authenticated using R'. Thus, summing the probabilities we find that the $\mathcal{A}$ has negligible advantage in winning the MA-security experiment:

$$\mathbf{Adv}_{G_4} = 0$$

We turn to bound the advantage of $\mathcal{A}$ in **Case 2.2**. **Case 2.2:** According to the definition of this case, UE accepts message $\mathbf{m_3}$ without a matching subset (i.e. without honest CN partner). In this subcase, $\mathcal{A}$ cannot corrupt the UE symmetric secret key (K).

Game 0: This is the original mutual authentication game described in 7.3:

$$\mathbf{Adv}^{\mathrm{MA,clean,C2}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(i,s) \in n_P \times n_S$ of the first UE session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index $(j) \in n_S$ of the first CN session that accepts without a matching subset, introducing a factor of n_S in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_1} \leq n_S \mathbf{Adv}_{G_2}.$$

Game 3 : In this game, we replace the computation of rk in π_r^j and π_t^{CN} sessions with uniformly random value ($\hat{rk}$). We do so by introducing a reduction $\mathcal{B}_1$, which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a PKW$^+$ challenger C_{PKW^+}. Every time $\mathcal{A}$ calls New(), $\mathcal{B}_1$ calls to PKW$^+$.New(), then we assign i to each $SUTI$ and a table $(SUTI, i)$ is maintained.

Additionally, $\mathcal{B}_1$ maintains the following registers C_{txt}, R_k and T_p. C_{txt} will be updated whenever we call $\mathsf{Wrap}(T, AD, m, i) \rightarrow C$ on a new entry, such that $(\cdot, T, AD, m, i) \notin C_{txt}$. Similarly, the R_k register will be updated whenever we call $\mathsf{Drv}(T, AD, i) \rightarrow rk$ on new entries, such that $(\cdot, T, AD, i) \notin R_k$. Finally, T_p will be updated whenever we call $\mathsf{Punc}(T, i, \rho)$.

Whenever **Send**(i, s, m, ρ) is called, we need first to check the T_p register and follow these conditions:

- IF $(T, i, \rho) \in T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0\|C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0\|C_1, SUTI||\Delta)$ by $\mathcal{B}_1$. and if Unwrap was queried, the computation of $m \leftarrow \mathsf{Unwrap}(K, T, C_0\|C_1, C_2)$ is replaced with a call to Unwrap $(i, T, C_0\|C_1, C_2)$ by $\mathcal{B}_1$. Similarly, whenever $\mathsf{Drv}(K, T, C_0\|C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0\|C_1)$.
- IF $(T, i, \overline{\rho} \in T_p$:Depending on the algorithm that was called, we check its registry for an output. If Unwrap or Drv were queried, we recover the output from the register (C_{txt}, R_k), respectively. However, if Wrap was queried, then the adversary has either issued a Corrupt, hence $\mathcal{B}_1$ can simulate Wrap, or the adversary has forged a message between UE and CN, hence breaking the PKW$^+$ security.
- IF $(T, i, \cdot) \notin T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0\|C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0\|C_1, SUTI||\Delta)$ by $\mathcal{B}_1$. Then, the output is added to the register as $(C, i, T, C_0\|C_1, SUTI||\Delta) \rightarrow C_{txt}$. Similarly, whenever $\mathsf{Drv}(K, T, C_0\|C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0\|C_1)$. and adds it to the register as $\mathsf{Drv}(i, T, C_0\|C_1) \rightarrow R_K$.

We note that this process is precisely how all parties interact with their collective PKW$^+$ state, making this replacement sound. If $\mathcal{A}$ can distinguish between these two games, then $\mathcal{A}$ breaks the PKW$^+$ key indistinguishability game. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{B}_1,\mathsf{PKW}^+}.$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_2 that decrypts correctly and is accepted by π_t^{CN}, but was not generated by π_r^j, breaking the authentication security of AEAD in PKW$^+$. In this game, no replacement is required because if the $\mathcal{A}$ could forge a cipher text, then they would win **Game 5**. Note that $(C_0\|C_1)$ is securely authenticated by PKW$^+$.Wrap() as additional data, and by the definition of **Case 1** $\mathcal{A}$ cannot issue **CorruptLTK**(UE). Any $\mathcal{A}$ that can trigger this abort event can be used to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{auth-aead}_{\mathsf{PKW}^+}.$$

Game 5: In this game we replace the authentication and mac keys AK, MK with uniformly random values $\hat{AK}, \hat{MK}$. We do so by defining a reduction a $\mathcal{B}_2$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{rk}$ and replacing the computation of AK, MK in any session that computes AK, MK, with the outputs from the C_{KDF} $\hat{AK}, \hat{MK}$. Since $AK, MK \leftarrow \mathsf{KDF}(rk, R)$ and by **Game 3** $\hat{rk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of KDF. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_2,\mathsf{KDF}}.$$

Game 6: In this game, we introduce an abort event that triggers if the same hash value exists for different inputs, thereby introducing hash collisions during the challenger execution with an honest session. We do so by defining a reduction $\mathcal{B}_3$ that initialises an

Hash challenger C_{Hash} and maintains a lookup table. Whenever $\mathcal{B}_3$ needs to hash a value, he queries the lookup table on x and recovers (out). The abort event only triggers if $\mathcal{A}$ can get the same output hash values from two distinct input values, thus finding a collision and breaking the collision security of the Hash scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of Hash. This implies:

$$\mathbf{Adv}_{G_5} \leq \mathbf{Adv}_{G_6} + \mathbf{Adv}^{collision}_{\mathcal{B}_3,\mathsf{Hash}}.$$

Game 7: In this game, we replace the response and session keys RK, SK with uniformly random values $\hat{RK}, \hat{SK}$. We do so by defining a reduction a $\mathcal{B}_4$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{AK}$ and replacing the computation of RK, SK in any session that computes RK, SK, with the outputs from the C_{KDF} $\hat{RK}, \hat{SK}$. Since $RK, SK \leftarrow \mathsf{KDF}(\hat{AK}, SUTI)$ and by **Game 5**,$\hat{AK}$ is already uniformly random, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 6** from **Game 7** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_6} \leq \mathbf{Adv}_{G_7} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Game 8: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a valid tag (MAC), but (MAC) was not output by π^t_j. We do so by defining a reduction $\mathcal{B}_5$ that initialises an MAC challenger C_{MAC}, which $\mathcal{B}_5$ queries when π^t_j needs to MAC (SQN, R) with $(\hat{MK})$ and from **Game 5** $\hat{MK}$ is already uniformly random and independent. This abort event occurs if MAC is verified correctly under $\hat{MK}$, but MAC was never produced by the C_{MAC}, breaking the security of the MAC scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the existential unforgeability security of the MAC scheme. This implies:

$$\mathbf{Adv}_{G_7} \leq \mathbf{Adv}_{G_8} + \mathbf{Adv}^{euf-cma}_{\mathcal{B}_5,\mathsf{MAC}}.$$

Game 9: In this game, we replace the response (RES^*) with uniformly random value $(\hat{RES^*})$. We do so by defining a reduction $\mathcal{B}_6$ that interacts with a KDF challenger C_{KDF}, querying C_{KDF} with $\hat{SK}, \hat{RK}$ and replacing the computation of (RES^*) in any session that computes RES^*, with the outputs from the $(C_{\mathsf{KDF}}$ $\hat{RK})$. Since $(RES^* \leftarrow \mathsf{KDF}(\hat{RK}, R\|\mathrm{gNB}_{name}))$ and by **Game 7** $\hat{RK}$ is already uniformly random, so these changes are sound. Any $\mathcal{A}$ that can distinguish **Game 8** from **Game 9** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_8} \leq \mathbf{Adv}_{G_9} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_6,\mathsf{KDF}}.$$

Game 10: In this game, we replace the (K_{SEAF}, HK) with uniformly random value $(\hat{K_{SEAF}}, \hat{HK})$. We do so by defining a reduction $\mathcal{B}_7$ that interacts with a KDF challenger C_{KDF}, querying C_{KDF} with $\hat{SK}$ and replacing the computation of (K_{SEAF}, HK) in any session that computes K_{SEAF}, HK, with the outputs from the $(C_{\mathsf{KDF}}$ $\hat{K_{SEAF}}\hat{HK})$. Since $(C_{\mathsf{KDF}}$ $\hat{K_{SEAF}}\hat{HK})$ and by **Game 7** $\hat{SK}$ is already uniformly random, so these changes are sound. Any $\mathcal{A}$ that can distinguish **Game 9** from **Game 10** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_9} \leq \mathbf{Adv}_{G_{10}} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_7,\mathsf{KDF}}.$$

By this stage, it is evident that the adversary is incapable of tampering with the messages exchanged between π^s_i and π^t_j sessions. In particular, the test session only accepts message m_3 from an honest matching partner because the $\mathcal{A}$ cannot modify the content of $AUTN$ without compromising the security of AEAD, KDF, MAC,and PKW^+. Furthermore, if the $\mathcal{A}$ attempts to alter R', the π^s_i will reject it since the decryption of R' is authenticated using MAC. This holds true since the adversary cannot corrupt the long-term UE symmetric-key, as per the definition of this case. Thus, summing the probabilities we find that the $\mathcal{A}$ has negligible advantage in winning the MA-security experiment:

$$\mathbf{Adv}_{G_{10}} = 0$$

Case 3: According to the definition of this case, CN accepts messages $\mathbf{m_1}$ or $\mathbf{m_5}$, without a matching subset (i.e. without honest UE partner).

Game 0: This is the original mutual authentication game described in 7.3:

$$\mathbf{Adv}^{\mathrm{MA,clean,C3}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(s) \in n_S$ of the first CN session that accepts without a matching subset, introducing a factor of n_S in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_S \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index $(t, j) \in n_P \times n_S$ of the first UE session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_1} \leq n_P \cdot n_S \mathbf{Adv}_{G_2}.$$

Game 3 : In this game, we replace the computation of rk in π^j_r and π^{CN}_t sessions with uniformly random value $(\hat{rk})$. We do so by introducing a reduction $\mathcal{B}_1$, which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a PKW^+ challenger C_{PKW^+}. Every time $\mathcal{A}$ calls New(), $\mathcal{B}_1$ calls to PKW^+.New(), then we assign i to each $SUTI$ and a table $(SUTI, i)$ is maintained. Additionally, $\mathcal{B}_1$ maintains the following registers C_{txt}, R_k and T_p. C_{txt} will be updated whenever we call $\mathsf{Wrap}(T, AD, m, i) \rightarrow C$ on a new entry, such that $(\cdot, T, AD, m, i) \notin C_{txt}$. Similarly, the R_k register will be updated whenever we call $\mathsf{Drv}(T, AD, i) \rightarrow rk$ on new entries, such that $(\cdot, T, AD, i) \notin R_k$. Finally, T_p will be updated whenever we call $\mathsf{Punc}(T, i, \rho)$. Whenever **Send**$(i, s, m, \rho)$ is called, we need first to check the T_p register and follow these conditions:

- IF $(T, i, \rho) \in T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0\|C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0\|C_1, SUTI||\Delta)$ by $\mathcal{B}_1$. and if Unwrap was queried, the computation of $m \leftarrow \mathsf{Unwrap}(K, T, C_0\|C_1, C_2)$ is replaced with a call to Unwrap $(i, T, C_0\|C_1, C_2)$ by $\mathcal{B}_1$. Similarly, whenever $\mathsf{Drv}(K, T, C_0\|C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0\|C_1)$.
- IF $(T, i, \overline{\rho} \in T_p$:Depending on the algorithm that was called, we check its registry for an output. If Unwrap or Drv were queried, we recover the output from the register (C_{txt}, R_k), respectively. However, if Wrap was queried, then the adversary has either issued a Corrupt, hence $\mathcal{B}_1$ can simulate Wrap, or the adversary has forged a message between UE and CN, hence breaking the PKW^+ security.
- IF $(T, i, \cdot) \notin T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0\|C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0\|C_1, SUTI||\Delta)$

by $\mathcal{B}_1$. Then, the output is added to the register as $(C, i, T, C_0\|C_1, SUTI\|\Delta) \rightarrow C_{txt}$. Similarly, whenever $\mathsf{Drv}(K, T, C_0\|C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0\|C_1)$. and adds it to the register as $\mathsf{Drv}(i, T, C_0\|C_1) \rightarrow R_K$.

We note that this process is precisely how all parties interact with their collective PKW$^+$ state, making this replacement sound. If $\mathcal{A}$ can distinguish between these two games, then $\mathcal{A}$ breaks the PKW$^+$ key indistinguishability game. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{B}_1,\mathsf{PKW}^+}.$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_2 that decrypts correctly and is accepted by π_t^{CN}, but was not generated by π_r^j, breaking the authentication security of AEAD in PKW$^+$. In this game, no replacement is required because if the $\mathcal{A}$ could forge a cipher text, then they would win **Game 5**. Note that $(C_0\|C_1)$ is securely authenticated by PKW$^+$.Wrap() as additional data, and by the definition of **Case 1** $\mathcal{A}$ cannot issue **CorruptLTK**(UE). Any $\mathcal{A}$ that can trigger this abort event can be used to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{auth-aead}_{\mathsf{PKW}^+}.$$

Game 5: In this game we replace the authentication and mac keys AK, MK with uniformly random values $\hat{AK}, \hat{MK}$. We do so by defining a reduction a $\mathcal{B}_2$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{rk}$ and replacing the computation of AK, MK in any session that computes AK, MK, with the outputs from the C_{KDF} $\hat{AK}, \hat{MK}$. Since $AK, MK \leftarrow \mathsf{KDF}(\hat{rk})$ and by **Game 5** $\hat{rk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of KDF. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_2,\mathsf{KDF}}.$$

Game 6: In this game, we introduce an abort event that triggers if the same hash value exists for different inputs, thereby introducing hash collisions during the challenger execution with an honest session. We do so by defining a reduction $\mathcal{B}_3$ that initialises an Hash challenger C_{Hash} and maintains a lookup table. Whenever $\mathcal{B}_3$ needs to hash a value, he queries the lookup table on x and recovers (out). The abort event only triggers if $\mathcal{A}$ can get the same output hash values from two distinct input values, thus finding a collision and breaking the collision security of the Hash scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of Hash. This implies:

$$\mathbf{Adv}_{G_6} \leq \mathbf{Adv}_{G_7} + \mathbf{Adv}^{collision}_{\mathcal{B}_3,\mathsf{Hash}}.$$

Game 8: In this game, we replace the response and session keys RK, SK with uniformly random values $\hat{RK}, \hat{SK}$. We do so by defining a reduction a $\mathcal{B}_4$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{AK}$ and replacing the computation of RK, SK in any session that computes RK, SK, with the outputs from the C_{KDF} $\hat{RK}, \hat{SK}$. Since $RK, SK \leftarrow \mathsf{KDF}(\hat{AK})$ and by **Game 5**,$\hat{AK}$ is already uniformly random, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 7** from **Game 8** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_7} \leq \mathbf{Adv}_{G_8} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Game 9: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a valid tag (MAC), but (MAC) was not output by π_j^t. We do so by defining a reduction $\mathcal{B}_5$ that initialises an MAC challenger C_{MAC}, which $\mathcal{B}_5$ queries when π_j^t needs to MAC (SQN, R) with $(\hat{MK})$ and from **Game 5** $\hat{MK}$ is already uniformly random and independent. This abort event occurs if MAC is verified correctly under $\hat{MK}$, but MAC was never produced by the C_{MAC}, breaking the security of the MAC scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the existential unforgeability security of the MAC scheme. This implies:

$$\mathbf{Adv}_{G_8} \leq \mathbf{Adv}_{G_9} + \mathbf{Adv}^{euf-cma}_{\mathcal{B}_5,\mathsf{MAC}}.$$

Game 10: In this game, we replace the response (RES^*) and (K_{SEAF}, HK) with uniformly random value $(\hat{RES^*})$ and $(\hat{K_{SEAF}}\hat{HK})$, respectively. We do so by defining a reduction $\mathcal{B}_6$ that interacts with a KDF challenger C_{KDF}, querying C_{KDF} with $\hat{SK}, \hat{RK}$ and replacing the computation of (RES^*) and (K_{SEAF}, HK) in any session that computes RES^*, K_{SEAF}, HK, with the outputs from the $(C_{\mathsf{KDF}}$ $\hat{RK})$ and $(C_{\mathsf{KDF}}$ $\hat{K_{SEAF}}\hat{HK})$. Since $(RES^* \leftarrow \mathsf{KDF}(\hat{RK}))$ and $(C_{\mathsf{KDF}}$ $\hat{K_{SEAF}}\hat{HK})$ and by **Game 8** $\hat{RK}$ and $\hat{SK}$ are already uniformly random, so these changes are sound. Any $\mathcal{A}$ that can distinguish **Game 9** from **Game 10** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_9} \leq \mathbf{Adv}_{G_{10}} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_6,\mathsf{KDF}}.$$

By this stage, it is evident that the adversary is incapable of tampering with the messages exchanged between π_i^s and π_j^t sessions. In particular, the test session only accepts message m_3 from an honest matching partner because the $\mathcal{A}$ cannot modify the content of $AUTN$ without compromising the security of AEAD, KDF, MAC,and PKW$^+$. Furthermore, if the $\mathcal{A}$ attempts to alter R', the π_i^s will reject it since the decryption of R' is authenticated using MAC. This holds true since the adversary cannot corrupt the long-term UE symmetric-key, as per the definition of this case. Thus, summing the probabilities we find that the $\mathcal{A}$ has negligible advantage in winning the MA-security experiment:

$$\mathbf{Adv}_{G_{10}} = 0$$

B.1.2 ***MA-security of PGUP HO (Proof of Theorem 2)***. Next, we show how the proposed handover protocol achieves mutual authentication. **Proof**: Our proof is divided into three cases, denoted by $\mathbf{Adv}^{\mathsf{MA,clean},c_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$, $\mathbf{Adv}^{\mathsf{MA,clean},c_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ and $\mathbf{Adv}^{\mathsf{MA,clean},c_3}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$

We then bound the advantage of $\mathcal{A}$ winning the game under certain assumptions to $\mathbf{Adv}^{\mathsf{MA,clean}_{HO}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq (\mathbf{Adv}^{\mathsf{MA,clean},c_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{MA,clean},c_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{MA,clean},c_3}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda))$.

- **Case 1:** assume that the first **clean** session π_i^s to accept without a matching session(i.e. $\pi_i^s.\rho$ = TgNB).
- **Case 2:** assume that the first **clean** session π_i^s to accept without a matching session(i.e. $\pi_i^s.\rho$ = UE).

Case 1: According to the definition of this case, TgNB accepts message$\mathbf{C}, \mathbf{C_{BS}}$ without an honest matching partner (SgNB) (no matching subset). In this case, the $\mathcal{A}$ cannot corrupt the Bootstrap-key (BK) between SgNB&TgNB.

Here we provide the security analysis:

Game 0: This is the original mutual authentication game described in 7.3:

$$\mathbf{Adv}^{\mathsf{MA,clean,C1}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(i, s) \in n_P \times n_S$ of the first TgNB session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \cdot \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index $j \in n_P$ of the *S*gNB party such that $\pi_i^s.pid = j$, introducing a factor of n_P in $\mathcal{A}$'s advantage.

$$\mathbf{Adv}_{G_1} \leq n_P \mathbf{Adv}_{G_2}.$$

Game 3 : In this game, we replace the computation of C with uniformly random value ($\hat{C}$). We do so by introducing a reduction $\mathcal{B}_1$, which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a PKW$^+$ challenger C_{PKW^+}. Every time $\mathcal{A}$ calls New(), $\mathcal{B}_1$ calls to PKW$^+$.New(), then we assign i to each $SUTI$ and a table $(SUTI, i)$ is maintained. Additionally, $\mathcal{B}_1$ maintains the following registers C_{txt}, R_k and T_p. C_{txt} will be updated whenever we call $\mathsf{Wrap}(T, AD, m, i) \rightarrow C$ on a new entry, such that $(\cdot, T, AD, m, i) \notin C_{txt}$. Similarly, T_p will be updated whenever we call $\mathsf{Punc}(T, i, \rho)$. Whenever **Send**$(i, s, m, \rho)$ is called, we need first to check the T_p register and follow these conditions:

- IF $(T, i, \rho) \in T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C \leftarrow \mathsf{Wrap}(BK, T, a, sk)$ is replaced with a call to $\mathsf{Wrap}(i, T, a, sk)$ by $\mathcal{B}_1$. and if Unwrap was queried, the computation of $m \leftarrow \mathsf{Unwrap}(BK, T, a, C)$ is replaced with a call to Unwrap (i, T, a, C) by $\mathcal{B}_1$. Similarly, whenever $\mathsf{Drv}(BK, T, C_0 \| C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0 \| C_1)$.
- IF $(T, i, \overline{\rho} \in T_p$:Depending on the algorithm that was called, we check its registry for an output. If Unwrap was queried, we recover the output from the register (C_{txt}). However, if Wrap was queried, then the adversary has either issued a Corrupt, hence $\mathcal{B}_1$ can simulate Wrap, or the adversary has forged a message between UE and CN, hence breaking the PKW$^+$ security.
- IF $(T, i, \cdot) \notin T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, a, sk)$ is replaced with a call to $\mathsf{Wrap}(i, T, a, sk)$ by $\mathcal{B}_1$. Then, the output is added to the register as $(C, i, T, a, sk) \rightarrow C_{txt}$.

We note that this process is precisely how all parties interact with their collective PKW$^+$ state, making this replacement sound. If $\mathcal{A}$ can distinguish between these two games, then $\mathcal{A}$ breaks the PKW$^+$ key indistinguishability game. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{B}_1, \mathsf{PKW}^+}.$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_{BS} that decrypts correctly and is accepted by π_i^s, but was not generated by π_t^j, breaking the authentication security of AEAD in PKW$^+$. In this game, no replacement is required because if the $\mathcal{A}$ could forge a cipher text, then they would win the game. Note that sk is randomly generated at the beginning of the session and C_{BS} is securely authenticated by PKW$^+$.Wrap. Any $\mathcal{A}$ that can trigger this abort event can be used to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{auth-aead}_{\mathsf{AEAD}}.$$

By this stage, it is evident that the adversary is incapable of tampering with the messages exchanged between π_i^s and π_j^t sessions. In particular, the test session only accepts message C, C_{BS} from an honest matching partner because the $\mathcal{A}$ cannot modify the content of either ciphertext without compromising the security of AEAD,and PKW$^+$. Thus, summing the probabilities we find that the $\mathcal{A}$ has negligible advantage in winning the MA-security experiment:

$$\mathbf{Adv}_{G_4} = 0$$

We now bound the advantage of $\mathcal{A}$ in **Case 2**. In this case, UE accepts message$\mathbf{C}_{\mathbf{UE}}$ without an honest matching partner (UE) (no matching subset). In this case, the $\mathcal{A}$ cannot corrupt the HO-key (HK) between *S*gNB&UE. Here we provide the security analysis:

Game 0: This is the original mutual authentication game described in 7.3:

$$\mathbf{Adv}^{\mathsf{MA,clean,C1}}_{\Pi, n_P, n_S, \mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(i, s) \in n_P \times n_S$ of the first TgNB session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \cdot \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index $j \in n_P$ of the *S*gNB party such that $\pi_i^s.pid = j$, introducing a factor of n_P in $\mathcal{A}$'s advantage.

$$\mathbf{Adv}_{G_1} \leq n_P \mathbf{Adv}_{G_2}.$$

Game 3: In this game, we replace ek, tk with uniformly random values $\hat{ek}, \hat{tk}$. We do so by defining a reduction a $\mathcal{B}_1$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{HK}$ and replacing the computation of ek, tk in any session that computes ek, tk, with the outputs from the C_{KDF} $\hat{ek}, \hat{tk}$. Since $ek, tk \leftarrow \mathsf{KDF}(\hat{HK})$ and by the definition of the execution environment, HK is generated uniformly random and independent, and by the definition of **Case 2** $\mathcal{A}$ cannot corrupt HK so this replacement is sound. Any $\mathcal{A}$ that can distinguish **Game 2** from **Game 3** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_1, \mathsf{KDF}}.$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_{UE} (keyed by $\hat{ek}$) that decrypts correctly and was not generated by *S*gNB, breaking the authentication security of AEAD. We do so by defining a reduction $\mathcal{B}_2$ that initialises an AEAD challenger C_{AEAD}, which $\mathcal{B}_2$ queries when he needs to encrypt/decrypt with $\hat{ek}$. By **Game 3** $\hat{ek}$ is already uniformly random and independent, and this replacement is undetectable. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_2$ to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{auth-aead}_{\mathcal{B}_2, \mathsf{AEAD}}.$$

Game 5: In this game, we replace handover keys HK^* with uniformly random values $\hat{HK^*}$. We do so by defining a reduction a $\mathcal{B}_3$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{tk}$ and replacing the computation of HK^* in any session that computes HK^*, with the outputs from the C_{KDF} $\hat{HK^*}$. Since $HK^* \leftarrow \mathsf{KDF}(\hat{tk})$ and by **Game 3** $\hat{tk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 4** from

Game 5 can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_3,\mathsf{KDF}}.$$

Game 6: In this game, we replace ck with uniformly random value $\hat{ck}$. We do so by defining a reduction a $\mathcal{B}_4$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $c\hat{k}^*$ and replacing the computation of ck in any session that computes ck, with the outputs from the C_{KDF} $H\hat{K}^*$. Since $ck \leftarrow \mathsf{KDF}(H\hat{K}^*)$ and by **Game 5** $H\hat{K}^*$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 5** from **Game 6** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_5} \leq \mathbf{Adv}_{G_6} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Game 7: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext HO_{res} (keyed by $\hat{ck}$) that decrypts correctly and was not generated by TgNB, breaking the authentication security of AEAD. We do so by defining a reduction $\mathcal{B}_5$ that initialises an AEAD challenger C_{AEAD}, which $\mathcal{B}_5$ queries when he needs to encrypt/decrypt with $\hat{ck}$. By **Game 6** $\hat{ck}$ is already uniformly random and independent, and this replacement is undetectable. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_6} \leq \mathbf{Adv}_{G_7} + \mathbf{Adv}^{auth-aead}_{\mathsf{AEAD}}$$

In this stage, it becomes evident that the $\mathcal{A}$ is unable to tamper with messages exchanged between the π^s_i and π^t_j. Specifically, π^s_i exclusively accepts message $\mathbf{C_{UE}}, \mathbf{HO_{res}}$ from an honest matching partner, and $\mathcal{A}$ cannot modify the content of either without compromising the security of PKW$^+$, AEAD&KDF. Thus, summing the probabilities we find that the $\mathcal{A}$ has negligible advantage in winning the MA-security experiment:

$$\mathbf{Adv}_{G_7} = 0$$

B.2 Key Indistinguishability- (Proof of Theorem 3)

In this section, we establish and demonstrate the key indistinguishability of our protocols.

B.2.1 KIND*-security of PGUP AKA.* In this subsection, we conduct a formal analysis of the KIND security of the initial authentication protocol. We have divided our proof into two cases: **Proof**:

1. **Case 1:** The test session π^s_i accepts messages without an honest matching session.
2. **Case 2:** The test session π^s_i accepts messages with a matching session.

We then bound the advantage of $\mathcal{A}$ winning the game subject to certain assumptions to $\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq (\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{IA},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{IA},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$.

Case 1: Here we present the analysis of **Case 1**, where π^s_i accepts without a matching session.

Game 0: This is the original key indistinguishability experiment described in 7.4:

$$\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{IA},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq Adv_{G_0}.$$

Game 1: In this game, we incorporate an abort event triggered by $\mathcal{A}$ issuing a query $\mathbf{Test}(i,s,\rho)$ where π^s_i accepts without a matching session. This scenario precisely mirrors the MA security experiment, and thus we have :

$$\mathbf{Adv}_{G_0} \leq \mathbf{Adv}_{G_1} + \mathbf{Adv}^{\mathsf{MA},\mathbf{clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda).$$

Since, as per **Case 1**, π^s_i has no matching session, and following the logic of **Game 1** where we abort upon π^s_i accepting without a matching session, it can be deduced that $\mathcal{A}$ is incapable of querying $\mathbf{Test}(i,s,\rho)$. Consequently, the KIND game unfolds identically, regardless of the bit b sampled by C. Thus

$$\mathbf{Adv}_{G_1} = 0.$$

Case 2: Here we present the analysis of **Case 2**, where π^s_i accepts with a matching session.

First, we recall the cleanness predicate 10, which restricts $\mathcal{A}$ from issuing a $\mathbf{Reveal}(i,s,\pi^s_i.\rho)$ query to π^s_i (and to any session π^t_j such that π^t_j is a matching session with π^s_i). Additionally, $\mathcal{A}$ is prevented from issuing a $\mathbf{StateReveal}(i,s,\pi^s_i.\rho)$ query, or any query to any session π^t_j such that π^t_j is a matching session with π^s_i. We proceed through a series of games to illustrate the security property.

Game 0: This is the original key indistinguishability experiment described in 7.4:

$$\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{IA},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq Adv_{G_0}.$$

Game 1: In this game, we guess the index $(i,s) \in n_P \times n_S$, and abort if $\mathcal{A}$ issues a $\mathbf{Test}(i^*,s^*,\pi^{s^*}_{i^*})$ query such that $i \neq i^*$ and $s \neq s^*$. This introduces the following bound:

$$\mathbf{Adv}_{G_0} \leq n_S \cdot n_P \cdot \mathbf{Adv}_{G_1}.$$

Game 2: In this game, we guess the index $(j,t) \in n_P \times n_S$, and abort if π^t_j is not the matching subset of π^s_i, which must exist by **Case 2**. This introduces the following bound:

$$\mathbf{Adv}_{G_1} \leq n_S \cdot n_P \cdot \mathbf{Adv}_{G_2}.$$

Game 3 : In this game, we replace the computation of rk in any sessions computes it with uniformly random value $(\hat{rk})$. We do so by introducing a reduction $\mathcal{B}_1$, which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a PKW$^+$ challenger C_{PKW^+}. Every time $\mathcal{A}$ calls New(), $\mathcal{B}_1$ calls to PKW$^+$.New(), then we assign i to each $SUTI$ and a table $(SUTI, i)$ is maintained. Additionally, $\mathcal{B}_1$ maintains the following registers C_{txt}, R_k and T_p. C_{txt} will be updated whenever we call $\mathsf{Wrap}(T,AD,m,i) \rightarrow C$ on a new entry, such that $(\cdot,T,AD,m,i) \notin C_{txt}$. Similarly, the R_k register will be updated whenever we call $\mathsf{Drv}(T,AD,i) \rightarrow rk$ on new entries, such that $(\cdot,T,AD,i) \notin R_k$. Finally, T_p will be updated whenever we call $\mathsf{Punc}(T,i,\rho)$. Whenever $\mathbf{Send}(i,s,m,\rho)$ is called, we need first to check the T_p register and follow these conditions:

- IF $(T,i,\rho) \in T_p$: We forward the query to C_{PKW^+}. In case Drv was queried, the computation of $\mathsf{Drv}(K,T,C_0\|C_1) \rightarrow rk$ is replaced with a call to $\mathsf{Drv}(i,T,C_0\|C_1)$ by $\mathcal{B}_1$.
- IF $(T,i,\overline{\rho} \in T_p$: If Drv were queried, we recover the output from the register (R_k).
- IF $(T,i,\cdot) \notin T_p$: We forward the query to C_{PKW^+}. In case Drv was queried, the computation of $\mathsf{Drv}(K,T,C_0\|C_1) \rightarrow rk$ is replaced with $\mathsf{Drv}(i,T,C_0\|C_1)$,by $\mathcal{B}_1$ and adds it to the register as $\mathsf{Drv}(i,T,C_0\|C_1) \rightarrow R_K$.

We note that this process is precisely how all parties interact with their collective PKW$^+$ state, making this replacement sound. If $\mathcal{A}$ can distinguish between these two games, then $\mathcal{A}$ breaks the PKW$^+$ key indistinguishability game. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{B}_1,\mathsf{PKW}^+}.$$

Game 4: In this game we replace the authentication and mac keys AK, MK with uniformly random values $\hat{AK}, \hat{MK}$. We do so by defining a reduction a $\mathcal{B}_2$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{rk}$ and replacing the computation of AK, MK in any session that computes AK, MK, with the outputs from the C_{KDF} $\hat{AK}, \hat{MK}$. Since $AK, MK \leftarrow \mathsf{KDF}(\hat{rk})$ and by **Game 3** $\hat{rk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_2$ to break the security of KDF. Thus:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_2,\mathsf{KDF}}.$$

Game 5: In this game, we replace the response and session keys RK, SK with uniformly random values $\hat{RK}, \hat{SK}$. We do so by defining a reduction a $\mathcal{B}_3$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{AK}$ and replacing the computation of RK, SK in π^s_i and π^t_j with the outputs from the C_{KDF} $\hat{RK}, \hat{SK}$. Since $RK, SK \leftarrow \mathsf{KDF}(\hat{AK})$ and by **Game 4**,$\hat{AK}$ is already uniformly random, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 4** from **Game 5** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_3,\mathsf{KDF}}.$$

Game 6: In this game, we replace the response (K_{SEAF}, HK) with uniformly random value $(\hat{K_{SEAF}}\hat{HK})$. We do so by reusing the reduction $\mathcal{B}_4$ that interacts with a KDF challenger C_{KDF}, querying C_{KDF} with $\hat{SK}, \hat{RK}$ and replacing the computation of (K_{SEAF}, HK) in π^s_i and π^t_j with the outputs from the (C_{KDF} $\hat{RK}$) and (C_{KDF} $\hat{K_{SEAF}}\hat{HK}$). Since $(K_{SEAF}, HK \leftarrow \mathsf{KDF}(\hat{SK}))$ and by **Game 5** $\hat{SK}$ is already uniformly random, so this change is sound. Any $\mathcal{A}$ that can distinguish **Game 5** from **Game 6** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_5} \leq \mathbf{Adv}_{G_6} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Here we emphasise that as a result of these changes, the session keys $\hat{rk}, \hat{AK}, \hat{MK}, \hat{RK}, \hat{SK}, \hat{K_{SEAF}}, \hat{HK}$ are now uniformly random and independent of the protocol execution regardless of the bit b sampled by C, thus $\mathcal{A}$ has no advantage in guessing the bit b:

$$\mathbf{Adv}_{G_6} = 0.$$

B.2.2 KIND-*security of Handover protocol.* Here we formally analyse the KIND-security of the HO protocol.

Proof: Our proof is divided into two cases, denoted by $\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ and $\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$

1. **Case 1:** The test session π^s_i accepts messages without a matching subset.
2. **Case 2:** The test session π^s_i accepts messages with a matching subset.

We then bound the advantage of $\mathcal{A}$ winning the game under certain assumptions to $\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{HO}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq (\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{KIND},\mathbf{clean},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$.

We begin by treating **Case 1**.

Case 1: Here we present the analysis of **Case 1**, where π^s_i accepts without a matching session.

Game 0: This is the original key indistinguishability experiment described in 7.4:

$$\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{HO},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq Adv_{G_0}.$$

Game 1: In this game, we incorporate an abort event triggered by $\mathcal{A}$ issuing a query $\mathbf{Test}(i, s, \rho)$ where π^s_i accepts without a matching session. This scenario precisely mirrors the MA security experiment, and thus we have :

$$\mathbf{Adv}_{G_0} \leq \mathbf{Adv}_{G_1} + \mathbf{Adv}^{\mathsf{MA},\mathbf{clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda).$$

Since, as per **Case 1**, π^s_i has no matching session, and following the logic of **Game 1** where we abort upon π^s_i accepting without a matching session, it can be deduced that $\mathcal{A}$ is incapable of querying $\mathbf{Test}(i, s, \rho)$. Consequently, the KIND game unfolds identically, regardless of the bit b sampled by C. Thus

$$\mathbf{Adv}_{G_1} = 0.$$

Case 2: Here we present the analysis of **Case 2**, where π^s_i accepts with a matching session.

First, we recall the cleanness predicate 10, which restricts $\mathcal{A}$ from issuing a $\mathbf{Reveal}(i, s, \pi^s_i.\rho)$ query to π^s_i (and to any session π^t_j such that π^t_j is a matching session with π^s_i). Additionally, $\mathcal{A}$ is prevented from issuing a $\mathbf{StateReveal}(i, s, \pi^s_i.\rho)$ query, or any query to any session π^t_j such that π^t_j is a matching session with π^s_i. We proceed through a series of games to illustrate the security property.

Game 0: This is the original key indistinguishability experiment described in 7.4:

$$\mathbf{Adv}^{\mathsf{KIND},\mathbf{clean}_{HO},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq Adv_{G_0}.$$

Game 1: In this game, we guess the index $(i, s) \in n_P \times n_S$, and abort if $\mathcal{A}$ issues a $\mathbf{Test}(i^*, s^*, \pi^{s^*}_{i^*})$ query such that $i \neq i^*$ and $s \neq s^*$. This introduces the following bound:

$$\mathbf{Adv}_{G_0} \leq n_S \cdot n_P \cdot \mathbf{Adv}_{G_1}.$$

Game 2: In this game, we guess the index $(j, t) \in n_P \times n_S$, and abort if π^t_j is not the matching subset of π^s_i, which must exist by **Case 2**. This introduces the following bound:

$$\mathbf{Adv}_{G_1} \leq n_S \cdot n_P \cdot \mathbf{Adv}_{G_2}.$$

Game 3: In this game, we replace ek, tk with uniformly random values $\hat{ek}, \hat{tk}$. We do so by defining a reduction a $\mathcal{B}_1$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{HK}$ and replacing the computation of ek, tk in any session that computes ek, tk, with the outputs from the C_{KDF} $\hat{ek}, \hat{tk}$. Since $ek, tk \leftarrow \mathsf{KDF}(\hat{HK})$ and by the definition of the execution environment, HK is generated uniformly random and independent, and by the definition of **Case 2** $\mathcal{A}$ cannot corrupt HK so this replacement is sound. Any $\mathcal{A}$ that can distinguish **Game 2** from **Game 3** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_1,\mathsf{KDF}}.$$

Game 4: In this game, we replace handover keys HK^* with uniformly random values $\hat{HK^*}$. We do so by defining a reduction a $\mathcal{B}_2$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{tk}$ and replacing the computation of HK^* in any session that computes

HK^*, with the outputs from the C_{KDF} $H\hat{K}^*$. Since $HK^* \leftarrow \mathsf{KDF}(\hat{tk})$ and by **Game 3** $\hat{tk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 3** from **Game 4** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_2,\mathsf{KDF}}.$$

Game 5: In this game, we replace $ck, \widehat{HK}$ with uniformly random value $\hat{ck}$. We do so by defining a reduction a $\mathcal{B}_3$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $H\hat{K}^*$ and replacing the computation of $ck, \widehat{HK}$ in any session that computes them, with the outputs from the C_{KDF} $H\hat{K}^*$. Since $ck, \widehat{HK} \leftarrow \mathsf{KDF}(H\hat{K}^*)$ and by **Game 4** $H\hat{K}^*$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 4** from **Game 5** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_3,\mathsf{KDF}}.$$

Here we emphasise that as a result of these changes, the session keys $\hat{ek}, \hat{tk}, H\hat{K}^*, \hat{ck}, \widehat{HK}$ are now uniformly random and independent of the protocol execution regardless of the bit b sampled by C; thus $\mathcal{A}$ has no advantage in guessing the bit b:

$$\mathbf{Adv}_{G_5} = 0.$$

B.3 Unlinkability (Proof of Theorem 4)

In this section, we establish and demonstrate the Unlinkability of PGUP scheme.

B.3.1 Unlink*-security of PGUP AKA.* In this subsection, we conduct a formal analysis of the Unlink security of the initial authentication protocol.

Our proof is divided into two cases, denoted by $\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{IA},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ and $\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{IA},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$

1. **Case 1:** The test session π_b (such that $\mathcal{A}$ issues $\mathbf{Test}(i,s,i^*,s^*)$) accepts messages without a matching subset.
2. **Case 2:** The test session π_b (such that $\mathcal{A}$ issues $\mathbf{Test}(i,s,i^*,s^*)$) accepts messages with a matching subset.

We then bound the advantage of $\mathcal{A}$ winning the game under certain assumptions to $\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq (\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$.

We begin by treating **Case 1**.

Case 1: π_b accepts without a matching subset. Here we describe the analysis of **Case 1**: **Game 0**: This is the original unlinkability experiment described in 7.5:

$$\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{IA},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq Adv_{G_0}.$$

Game 1: In this game, we introduce an abort event that triggers if $\mathcal{A}$ issues a query $\mathbf{Test}(i,s,i^*,s^*)$ and π_b accepts without a matching session or subset. This is exactly equal to the MA security experiment, and thus we have :

$$\mathbf{Adv}_{G_0} \leq \mathbf{Adv}_{G_1} + \mathbf{Adv}^{\mathsf{MA},\mathbf{clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda).$$

Considering **Case 1**, where π lacks a matching session, and according to **Game 1**, if π_b accepts without such a match, we abort. Consequently, $\mathcal{A}$ is unable to terminate and produce a guess bit b'. Consequently, the Unlink game progresses uniformly irrespective of the bit b sampled by C. Thus,

$$\mathbf{Adv}_{G_1} = 0.$$

We now turn to **Case 2**.

Case 2: π^s_i accepts with a matching subset.

First, we recall that cleanness predicate Definition 11 prevents the $\mathcal{A}$ from issuing a $\mathbf{StateReveal}(i,s,\pi^s_i.\rho)$, nor to any session π^t_j such that π^t_j is a matching session or subset with π^s_i. We proceed via the following sequence of games.

Game 0: This is the original unlinkability experiment described in 7.5

$$\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{IA}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(i,s) \in n_P \times n_S$ of the first session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index $(t,j) \in n_P \times n_S$ of the first UE session that accepts without a matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_1} \leq n_P \cdot n_S \mathbf{Adv}_{G_2}.$$

Game 3 : In this game, we guess session index $r \in n_S$ and aborts if π^j_r registers a ciphertext in $C_{txt} \leftarrow (C,T,AD,m,j)$ and π^{CN}_t computes $rk \leftarrow \mathsf{Drv}(k_i,T',C_0)$ and $(.,T',.,.) \in C_{txt}$ but $T' \neq T$, introducing a factor of n_S in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_2} \leq n_S \mathbf{Adv}_{G_3}.$$

Game 4 : In this game, we replace the computation of rk in π^j_r and π^{CN}_t sessions with uniformly random value $(\hat{rk})$. We do so by introducing a reduction $\mathcal{B}_1$, which operates as follows: at the beginning of the game, $\mathcal{B}_1$ initiates a PKW$^+$ challenger C_{PKW^+}. Every time $\mathcal{A}$ calls New(), $\mathcal{B}_1$ calls to PKW$^+$.New(), then we assign i to each $SUTI$ and a table $(SUTI,i)$ is maintained.

Additionally, $\mathcal{B}_1$ maintains the following registers C_{txt}, R_k and T_p. C_{txt} will be updated whenever we call $\mathsf{Wrap}(T,AD,m,i) \rightarrow C$ on a new entry, such that $(\cdot,T,AD,m,i) \notin C_{txt}$. Similarly, the R_k register will be updated whenever we call $\mathsf{Drv}(T,AD,i) \rightarrow rk$ on new entries, such that $(\cdot,T,AD,i) \notin R_k$. Finally, T_p will be updated whenever we call $\mathsf{Punc}(T,i,\rho)$.

Whenever $\mathbf{Send}(i,s,m,\rho)$ is called, we need first to check the T_p register and follow these conditions:

- IF $(T,i,\rho) \in T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K,T,C_0\|C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i,T,C_0\|C_1,SUTI||\Delta)$ by $\mathcal{B}_1$. and if Unwrap was queried, the computation of $m \leftarrow \mathsf{Unwrap}(K,T,C_0\|C_1,C_2)$ is replaced with a call to Unwrap $(i,T,C_0\|C_1,C_2)$ by $\mathcal{B}_1$. Similarly, whenever $\mathsf{Drv}(K,T,C_0\|C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i,T,C_0\|C_1)$.
- IF $(T,i,\overline{\rho} \in T_p$:Depending on the algorithm that was called, we check its registry for an output. If Unwrap or Drv were queried, we recover the output from the register (C_{txt},R_k), respectively. However, if Wrap was queried, then the adversary has either issued a Corrupt, hence $\mathcal{B}_1$ can simulate Wrap, or the adversary has forged a message between UE and CN, hence breaking the PKW$^+$ security.

- IF $(T, i, \cdot) \notin T_p$: We forward the query to C_{PKW^+}. In case Wrap was queried, the computation of $C_2 \leftarrow \mathsf{Wrap}(K, T, C_0\|C_1, SUTI||\Delta)$ is replaced with a call to $\mathsf{Wrap}(i, T, C_0\|C_1, SUTI||\Delta)$ by $\mathcal{B}_1$. Then, the output is added to the register as $(C, i, T, C_0\|C_1, SUTI||\Delta) \rightarrow C_{txt}$. Similarly, whenever $\mathsf{Drv}(K, T, C_0\|C_1) \rightarrow rk$ is called, $\mathcal{B}_1$ replaces the computation with $\mathsf{Drv}(i, T, C_0\|C_1)$. and adds it to the register as $\mathsf{Drv}(i, T, C_0\|C_1) \rightarrow R_K$.

We note that this process is precisely how all parties interact with their collective PKW^+ state, making this replacement sound. If $\mathcal{A}$ can distinguish between these two games, then $\mathcal{A}$ breaks the PKW^+ key indistinguishability game. Thus:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{\mathsf{KIND}}_{\mathcal{B}_1,\mathsf{PKW}^+}.$$

Game 5: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_2 that decrypts correctly and is accepted by π_t^{CN}, but was not generated by π_r^j, breaking the authentication security of AEAD in PKW^+. In this game, no replacement is required because if the $\mathcal{A}$ could forge a cipher text, then they would win **Game 5**. Note that $(C_0\|C_1)$ is securely authenticated by PKW^+.Wrap() as additional data, and by the definition of **Case 1** $\mathcal{A}$ cannot issue **CorruptLTK**(UE). Any $\mathcal{A}$ that can trigger this abort event can be used to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{auth-aead}_{\mathsf{PKW}^+}.$$

Game 6: In this game we replace the authentication and mac keys AK, MK with uniformly random values $\hat{AK}, \hat{MK}$. We do so by defining a reduction a $\mathcal{B}_2$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{rk}$ and replacing the computation of AK, MK in any session that computes AK, MK, with the outputs from the C_{KDF} $\hat{AK}, \hat{MK}$. Since $AK, MK \leftarrow \mathsf{KDF}(rk)$ and by **Game 5** $\hat{rk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of KDF. Thus:

$$\mathbf{Adv}_{G_5} \leq \mathbf{Adv}_{G_6} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_2,\mathsf{KDF}}.$$

Game 7: In this game, we introduce an abort event that triggers if the same hash value exists for different inputs, thereby introducing hash collisions during the challenger execution with an honest session. We do so by defining a reduction $\mathcal{B}_3$ that initialises an Hash challenger C_{Hash} and maintains a lookup table. Whenever $\mathcal{B}_3$ needs to hash a value, he queries the lookup table on x and recovers (out). The abort event only triggers if $\mathcal{A}$ can get the same output hash values from two distinct input values, thus finding a collision and breaking the security of the Hash scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_3$ to break the security of Hash. This implies:

$$\mathbf{Adv}_{G_6} \leq \mathbf{Adv}_{G_7} + \mathbf{Adv}^{collision}_{\mathcal{B}_3,\mathsf{Hash}}.$$

Game 8: In this game, we replace the response and session keys RK, SK with uniformly random values $\hat{RK}, \hat{SK}$. We do so by defining a reduction a $\mathcal{B}_4$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{AK}$ and replacing the computation of RK, SK in any session that computes RK, SK, with the outputs from the C_{KDF} $\hat{RK}, \hat{SK}$. Since $RK, SK \leftarrow \mathsf{KDF}(\hat{AK})$ and by **Game 7**,$\hat{AK}$ is already uniformly random, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 8** from **Game 9** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_7} \leq \mathbf{Adv}_{G_8} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Game 9: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a valid tag (MAC), but (MAC) was not output by π_j^t. We do so by defining a reduction $\mathcal{B}_5$ that initialises an MAC challenger C_{MAC}, which $\mathcal{B}_5$ queries when π_j^t needs to MAC (SQN, R) with $(\hat{MK})$ and from **Game 7** $\hat{MK}$ is already uniformly random and independent. This abort event occurs if MAC is verified correctly under $\hat{MK}$, but MAC was never produced by the C_{MAC}, breaking the security of the MAC scheme. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the existential unforgeability security of the MAC scheme. This implies:

$$\mathbf{Adv}_{G_8} \leq \mathbf{Adv}_{G_9} + \mathbf{Adv}^{euf-cma}_{\mathcal{B}_5,\mathsf{MAC}}.$$

Game 10: In this game, we replace the response (RES^*) and (K_{SEAF}, HK) with uniformly random value $(\hat{RES^*})$ and $(\hat{K_{SEAF}}\hat{HK})$, respectively. We do so by defining a reduction $\mathcal{B}_6$ that interacts with a KDF challenger C_{KDF}, querying C_{KDF} with $\hat{SK}, \hat{RK}$ and replacing the computation of (RES^*) and (K_{SEAF}, HK) in any session that computes RES^*, K_{SEAF}, HK, with the outputs from the (C_{KDF} $\hat{RK}$) and (C_{KDF} $\hat{K_{SEAF}}\hat{HK}$). Since $(RES^* \leftarrow \mathsf{KDF}(\hat{RK}))$ and (C_{KDF} $\hat{K_{SEAF}}\hat{HK}$) and by **Game 9** $\hat{RK}$ and $\hat{SK}$ are already uniformly random, so these changes are sound. Any $\mathcal{A}$ that can distinguish **Game 9** from **Game 10** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_9} \leq \mathbf{Adv}_{G_{10}} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_6,\mathsf{KDF}}.$$

In this stage, it is important to highlight that all plaintext messages that are sent and received across the network to and from π_b and its corresponding session and subsets are uniformly random and independent of the bit b that is selected by the challenger. As a result, $\mathcal{A}$ has no advantage in predicting the bit b. By adding up the probabilities, it becomes evident that $\mathcal{A}$ has a negligible advantage in winning the Unlink game. Thus:

$$\mathbf{Adv}_{G_{10}} = 0$$

B.3.2 Unlink*-security of PGUP HO.* In this subsection, we conduct a formal analysis of the Unlink security of the HO protocol. We have divided our proof into two cases:

Our proof is divided into two cases, denoted by $\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{HO},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$ and $\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{HO},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$

1. **Case 1:** The test session π_b (such that $\mathcal{A}$ issues $\mathbf{Test}(i, s, i^*, s^*)$) accepts messages without a matching subset.
2. **Case 2:** The test session π_b (such that $\mathcal{A}$ issues $\mathbf{Test}(i, s, i^*, s^*)$) accepts messages with a matching subset.

We then bound the advantage of $\mathcal{A}$ winning the game under certain assumptions to $\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq (\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) + \mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean},C_2}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda)$.

We begin by treating **Case 1**.

Case 1: π_b accepts without a matching subset. Here we describe the analysis of **Case 1**: **Game 0**: This is the original unlinkability experiment described in 7.5:

$$\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{HO},C_1}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq Adv_{G_0}.$$

Game 1: In this game, we introduce an abort event that triggers if $\mathcal{A}$ issues a query $\mathbf{Test}(i, s, i^*, s^*)$ and π_b accepts without a matching session or subset. This is exactly equal to the MA security experiment, and thus we have :

$$\mathbf{Adv}_{G_0} \leq \mathbf{Adv}_{G_1} + \mathbf{Adv}^{\mathsf{MA},\mathbf{clean}_{HO}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda).$$

Considering **Case 1**, where π lacks a matching session, and according to **Game 1**, if π_b accepts without such a match, we abort. Consequently, $\mathcal{A}$ is unable to terminate and produce a guess bit b'. Consequently, the Unlink game progresses uniformly irrespective of the bit b sampled by C. Thus,

$$\mathbf{Adv}_{G_1} = 0.$$

We now turn to **Case 2**.

Case 2: π_i^s accepts with a matching subset.

First, we recall that cleanness predicate Definition 11 prevents the $\mathcal{A}$ from issuing a $\mathbf{StateReveal}(i, s, \pi_i^s.\rho)$, nor to any session π_j^t such that π_j^t is a matching session or subset with π_i^s. We proceed via the following sequence of games.

Game 0: This is the original unlinkability experiment described in 7.5

$$\mathbf{Adv}^{\mathsf{Unlink},\mathbf{clean}_{HO}}_{\Pi,n_P,n_S,\mathcal{A}}(\lambda) \leq \mathbf{Adv}_{G_0}.$$

Game 1 : In this game, we guess the index $(i, s) \in n_P \times n_S$ of the test session, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_0} \leq n_P \cdot n_S \mathbf{Adv}_{G_1}.$$

Game 2 : In this game, we guess the index $(t, j) \in n_P \times n_S$ of the test session's matching subset, introducing a factor of $n_P \times n_S$ in $\mathcal{A}$'s advantage:

$$\mathbf{Adv}_{G_1} \leq n_P \cdot n_S \mathbf{Adv}_{G_2}.$$

Game 3: In this game, we replace ek, tk with uniformly random values $\hat{ek}, \hat{tk}$. We do so by defining a reduction a $\mathcal{B}_1$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{HK}$ and replacing the computation of ek, tk in any session that computes ek, tk, with the outputs from the C_{KDF} $\hat{ek}, \hat{tk}$. Since $ek, tk \leftarrow \mathsf{KDF}(\hat{HK})$ and by the definition of the execution environment, HK is generated uniformly random and independent, and by the definition of **Case 2** $\mathcal{A}$ cannot corrupt HK so this replacement is sound. Any $\mathcal{A}$ that can distinguish **Game 2** from **Game 3** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_2} \leq \mathbf{Adv}_{G_3} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_1,\mathsf{KDF}}.$$

Game 4: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext C_{UE} (keyed by $\hat{ek}$) that decrypts correctly and was not generated by SgNB, breaking the authentication security of AEAD. We do so by defining a reduction $\mathcal{B}_2$ that initialises an AEAD challenger C_{AEAD}, which $\mathcal{B}_2$ queries when he needs to encrypt/decrypt with $\hat{ek}$. By **Game 3** $\hat{ek}$ is already uniformly random and independent, and this replacement is undetectable. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_2$ to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_3} \leq \mathbf{Adv}_{G_4} + \mathbf{Adv}^{auth-aead}_{\mathcal{B}_2,\mathsf{AEAD}}.$$

Game 5: In this game, we replace handover keys HK^* with uniformly random values $\hat{HK}^*$. We do so by defining a reduction a $\mathcal{B}_3$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{tk}$ and replacing the computation of HK^* in any session that computes HK^*, with the outputs from the C_{KDF} $\hat{HK}^*$. Since $HK^* \leftarrow \mathsf{KDF}(\hat{tk})$ and by **Game 3** $\hat{tk}$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 4** from **Game 5** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_4} \leq \mathbf{Adv}_{G_5} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_3,\mathsf{KDF}}.$$

Game 6: In this game, we replace ck with uniformly random value $\hat{ck}$. We do so by defining a reduction a $\mathcal{B}_4$, that initialises an KDF challenger C_{KDF}, querying C_{KDF} with $\hat{ck}^*$ and replacing the computation of ck in any session that computes ck, with the outputs from the C_{KDF} $\hat{HK}^*$. Since $ck \leftarrow \mathsf{KDF}(\hat{HK}^*)$ and by **Game 5** $\hat{HK}^*$ is already uniformly random and independent, this change is sound. Any $\mathcal{A}$ that can distinguish **Game 5** from **Game 6** can be used to break KDF security of the KDF scheme. Thus:

$$\mathbf{Adv}_{G_5} \leq \mathbf{Adv}_{G_6} + \mathbf{Adv}^{\mathsf{KDF}}_{\mathcal{B}_4,\mathsf{KDF}}.$$

Game 7: In this game, we introduce an abort event that triggers if $\mathcal{A}$ generates a ciphertext HO_{res} (keyed by $\hat{ck}$) that decrypts correctly and was not generated by TgNB, breaking the authentication security of AEAD. We do so by defining a reduction $\mathcal{B}_5$ that initialises an AEAD challenger C_{AEAD}, which $\mathcal{B}_5$ queries when he needs to encrypt/decrypt with $\hat{ck}$. By **Game 6** $\hat{ck}$ is already uniformly random and independent, and this replacement is undetectable. Any $\mathcal{A}$ that can trigger the abort event can be used by $\mathcal{B}_5$ to break the security of AEAD. This implies:

$$\mathbf{Adv}_{G_6} \leq \mathbf{Adv}_{G_7} + \mathbf{Adv}^{auth-aead}_{\mathsf{AEAD}}$$

In this stage, it is important to highlight that all plaintext messages that are sent and received across the network to and from π_b and its corresponding session and subsets are uniformly random and independent of the bit b that is selected by the challenger. As a result, $\mathcal{A}$ has no advantage in predicting the bit b. By adding up the probabilities, it becomes evident that $\mathcal{A}$ has a negligible advantage in winning the Unlink game. Thus:

$$\mathbf{Adv}_{G_7} = 0$$