Active Attack Resilience in 5G: A New Take on Authentication and Key Agreement

Nazatul H. Sultan*, Xinlong Guan*, Josef Pieprzyk*†, Wei Ni*, Sharif Abuadbba*, and Hajime Suzuki*
*CSIRO's Data61, Australia
†Institute of Computer Science, Polish Academy of Sciences, Poland

***Abstract*—As 5G networks continue to expand into critical infrastructure, ensuring secure and efficient user authentication has become more important than ever. The 5G-AKA protocol, standardized by 3GPP in TS 33.501, is the cornerstone of authentication in current 5G deployments. It provides mutual authentication, user privacy, and key secrecy. However, despite its widespread adoption, 5G-AKA suffers from known limitations in both security and performance. While it primarily focuses on protecting privacy against passive attackers, recent studies have highlighted its vulnerabilities to active attacks. Furthermore, it relies on a sequence number-based mechanism to prevent replay attacks, requiring the user device and the core network to remain perfectly synchronized. This stateful design introduces operational complexity, frequent desynchronization issues, and additional communication overhead. More critically, 5G-AKA lacks Perfect Forward Secrecy (PFS), leaving past communications vulnerable if long-term keys are ever compromised- a growing concern in the age of sophisticated adversaries.**

In this paper, we propose an enhanced authentication protocol that builds on the design principles of 5G-AKA while addressing these fundamental shortcomings. First, we present a stateless version of the protocol that eliminates the reliance on sequence numbers, reducing communication complexity while remaining fully compatible with existing SIM cards and network infrastructure. We then extend this design to include PFS with only minimal cryptographic overhead. Both protocols are rigorously analyzed using ProVerif, showing that they meet all major security requirements, including resistance to both passive and active attacks, as well as those outlined by 3GPP and recent academic studies. We also prototype both protocols and evaluate their performance against 5G-AKA and 5G-AKA′ (USENIX'21). Our results show that the proposed protocols offer stronger security guarantees with only minor impact on computational costs, making them practical and forward-compatible solutions for 5G and beyond.

***Index Terms*—5G; Authentication; Key Agreement; Privacy; Perfect Forward Secrecy**

I. Introduction

With recent advancements in Fifth-Generation (5G) mobile network technology, there has been a significant shift from 3G/4G to 5G. According to Ericsson, by the end of 2029, 5G is expected to cover approximately 85% of the world's population[1]. To ensure the security and privacy of 5G users, the Third Generation Partnership Project (3GPP)[2] has proposed various security standards. One such standard is the 5G-Authentication and Key Agreement (5G-AKA) protocol, detailed in 3GPP Technical Specification (TS) 33.501 [1].

[1]https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/network-coverage

[2]https://www.3gpp.org/

The 5G-AKA protocol involves three main entities: the Subscriber (or User Equipment, UE), the Serving Network (SN), and the Home Network (HN). The subscriber refers to the mobile user connected to the 5G network via a Universal Integrated Circuit Card (UICC), commonly known as a SIM card.[3] The SN and HN represent the base station of a network carrier and the subscriber's home carrier, respectively. 5G-AKA provides essential security features, including mutual authentication and the establishment of a secure session key between the UE and the network, thereby ensuring secure communication.

The 5G-AKA protocol builds on the earlier 3G/4G-AKA protocols, offering improved privacy through the use of public-key encryption to protect the subscriber's permanent identity (i.e., Subscription Permanent Identifier, SUPI) [2]. As a foundational component of 5G security, 5G-AKA has undergone extensive security evaluations [2]–[4], similar to its predecessors [5], [6]. These analyses have uncovered the following major inherent issues in the 5G-AKA protocol (we also discuss it in detail in Section III-A):

a) Lack of Active Attack Resistance: The 5G-AKA protocol is primarily designed to defend against passive attackers, who only eavesdrop on communications. However, recent studies [2], [7], [8] have shown that it remains vulnerable to active attackers, who can manipulate, replay, or inject messages into the protocol flow, thereby compromising user privacy. Prior research [5], [7]–[9] have also demonstrated privacy attacks—especially against subscriber unlinkability—that allow adversaries to distinguish and track specific users. These threats are particularly concerning for high-profile individuals (e.g., journalists, activists, or political figures). With the emergence of open-source 5G platforms such as [10]–[12], the feasibility of active attacks has become even more realistic [13].

b) Lack of Perfect Forward Secrecy (PFS): The 5G-AKA does not offer PFS [2]. If the UE's long-term secret key is compromised, an adversary can compute past session keys and decrypt previously captured traffic. This is particularly a serious concern in an era of advanced persistent threats where long-term key compromise is increasingly plausible.

c) Inefficient Replay Attack Prevention: The 5G-AKA protocol uses a sequence number-based mechanism to prevent replay attacks. This requires synchronization of the sequence

[3]We use "subscriber" and "UE" interchangeably throughout the paper.

number between the UE and HN, making the protocol stateful. This stateful design introduces additional challenges such as operational complexity, frequent desynchronization issues, and communication overhead for resynchronization [1], [2].

Furthermore, the comprehensive formal analysis by Basin *et al.* [2] (CCS'18), based on 3GPP's security specifications, also highlights several underspecified requirements and inherited weaknesses from 3G/4G-AKA, such as the lack of key confirmation. The authors recommend designing a protocol specifically tailored for 5G, rather than extending legacy systems, due to outdated assumptions and constraints (e.g., lack of robust pseudo-random number generators) that have since been addressed in modern 5G infrastructure.

To address these challenges, several studies—including [8], [13]–[15]—have proposed enhancements to the 5G-AKA protocol. However, most of these solutions fall short of delivering all essential security properties within a unified framework. In particular, they often fail to simultaneously defend against active attacks, provide PFS, and implement a robust replay protection mechanism that does not rely on sequence numbers—all while maintaining compatibility with existing 5G SIM cards. We discuss these works in more detail in the Related Work section.

Our Contribution: In this paper, we delve deeper into the issues present in the 5G-AKA protocol and propose an enhanced and secure version of the AKA protocol tailored for 5G. We summarize our key contributions below.

- We design an efficient and secure AKA protocol for 5G that fulfills all security guarantees outlined in 3GPP TS 33.501 [1]. Our protocol, referred to as Protocol I, also addresses the additional underspecified security requirements discussed in [2]. Importantly, our protocol is resilient against both passive and active attackers. While following a message flow pattern similar to 5G-AKA, we modify the challenge-response method to prevent replay attacks and ensure mutual authentication between the subscriber and the HN. This new method avoids the inefficient sequence number-based replay attack prevention techniques used in 5G-AKA while maintaining compatibility with the existing 5G infrastructure, especially 5G SIM cards.
- We propose an extension of our protocol I to accommodate the additional property of PFS with the expense of a few additional lightweight cryptographic operations. In our extension, referred to as Protocol II, we efficiently introduce an ephemeral Diffie-Hellman (DH) key exchange method along with a modified challenge-response process from Protocol I. This extension supports all essential security requirements of our protocol I, including resistance to both passive and active attacks.
- We perform a comprehensive formal security analysis using the state-of-the-art symbolic model-based automated security verification tool *ProVerif* [16]. Our formal analysis indicates both of our protocols (Protocol I and Protocol II) support all essential security requirements outlined in 3GPP TS 33.501 [1] and [2], including mutual authentication, secrecy, and active attack resistance. Additionally, our analysis shows that the extension protocol II offers PFS. All code is available at https://anonymous.4open.science/r/AKA-5G-ProVerif-9378/.
- We provide a comprehensive comparison of our protocols with the 5G-AKA protocol [1] and 5G-AKA′ protocol [13] (USENIX'21). Additionally, we conduct experiments using the Crypto++ library [17] to validate our findings. All code is available at https://anonymous.4open.science/r/AKA-5G-E8B4/.

TABLE I: Functionality and Security Comparison between Our Protocols and Other Notable Works

	Mutual Authentication	Active Attack Resistance	PFS	SQN Resync	Compatibility#	
					USIM	SN
DDAIP [18]	✓	✗	✗	✓	✓	✓
MultiIMSI-4G [6]	✓	✗	✗	✓	✓	✓
DefeatCatchers [19]	✓	✗	✗	✓	✓	✓
PrivacyThreats-5G [7]	✓	✗	✗	✓	✓	✓
AKA-FS [20]*	✓	✓	✓	✓	✗	✗
TSA-5G [21]*	✓	✓	✓	✓	✗	✗
5G-AKA-FS [22]*	✓	✓	✓	✓	✗	✗
Symmetric-AKA [14]	✓	✗	✓	–	✓	✓
Beyond-5G [23]	✓	✓	✓	–	✗	✗
AKA$^+$ [8]	✓	✓	✗	✓	✗	✗
5G-AKA′ [13]	✓	✓	✗	✓	✓	✓
5G-AKA [1]	✓	✗	✗	✓	✓	✓
Our Protocol I	✓	✓	✗	–	✓	✓
Our Protocol II	✓	✓	✓	–	✓	✓

✓indicates that the property is supported by the protocol; ✗indicates that the property is not supported by the protocol; ✓indicates the functionality is required by the protocol; – indicates the functionality is not required by the protocol; PFS means Perfect Forward Secrecy; SQN Resync means Sequence number re-synchronization process; * [20], [21], and [22] have not provided thorough formal security analyses; #: these properties have not yet been experimentally verified.

II. Related Work

The 5G-AKA protocol has undergone thorough scrutiny, revealing several limitations, particularly regarding subscribers' privacy. Several existing studies have identified these issues and proposed various mitigation techniques. In this section, we briefly introduce existing efforts aimed at enhancing the security and privacy of the 5G-AKA protocol, including those that have conducted formal security analyses. Table I presents a summarized comparison of the security and functionality between our protocols and existing notable works.

In [18], a pseudonym mechanism was proposed to mitigate linkability (or distinguishability) attacks on subscriber privacy in 5G. This mechanism allows subscribers to utilize multiple identities instead of a single permanent identity (i.e., SUPI). A similar pseudonym-based mechanism was also proposed for 3G/4G-AKA protocols in [6], [19]. While this technique enhances subscriber privacy to some extent, it does not offer protection against active attackers [13]. In [7], an attack was presented that could disclose the sequence numbers of targeted subscribers. This attack exploits weaknesses in the sequence number protection mechanism in the 5G-AKA. Although three countermeasures were proposed, unfortunately, none of them can prevent the encrypted SUPI replay attack demonstrated in [9], [8].

In [8], the AKA$^+$ scheme was proposed for 5G, aiming to withstand all known types of attacks on subscribers' privacy. It implements two key strategies to mitigate privacy breaches in the 5G-AKA protocol. Firstly, it delays the re-synchronization

message, which is utilized when the subscriber and the HN become unsynchronized regarding the common sequence number. Secondly, it introduces a challenge message to the subscriber before initiating authentication. However, AKA$^+$ brings about several significant modifications to the 5G-AKA protocol. Most notably, it renders the existing USIM (Universal Subscriber Identity Module) commands incompatible with AKA$^+$, necessitating the replacement of USIM cards with new ones [13].

In [13], the 5G-AKA$'$ was proposed with the aim of mitigating all known privacy-related attacks in 5G-AKA while making minimal modifications. The key idea is to verify the freshness of the challenge message from the HN before the subscriber processes other operations. Unlike AKA$^+$, 5G-AKA$'$ is compatible with existing USIM cards. However, 5G-AKA$'$ inherits other limitations of the 5G-AKA protocol, including sequence number resynchronization and the lack of forward secrecy if the long-term secret key of the subscriber and the private key of the HN are compromised. In [20], [21], and [22], three DH key exchange-based schemes have been proposed. However, similar to AKA$^+$, both [20] and [21] are also incompatible with existing USIM cards and SNs' implementations. Further, the works [20], [21], and [22] have not provided thorough security analyses. This means that the reader cannot be sure whether or not the claimed security goals are indeed achieved.

In [14], a symmetric key-based AKA protocol is proposed for 5G. The protocol does not use any public-key cryptographic primitives and supports some of the essential security requirements such as anonymity, unlinkability, mutual authentication, and confidentiality. However, the work [14] is vulnerable to location confidentiality attacks, as shown in [15]. In [23], a quantum-safe AKA protocol is proposed for 5G. The scheme uses post-quantum KEM (Key Encapsulation Mechanism) instead of relying on the public-key cryptographic primitives. It is incompatible with existing USIM cards and SNs' implementations, similar to [8], [20], and [21].

Apart from proposing new schemes to enhance 5G-AKA protocols, extensive security analyses have been conducted on the existing 5G-AKA protocol. The authors in [2] provide a detailed formal security analysis of 5G-AKA, offering a comprehensive definition of its security and privacy properties based on 3GPP's specifications. Utilizing Tamarin Prover [24], the analysis identifies several underspecified security requirements within the 5G-AKA protocol, including the lack of forward secrecy, vulnerability to active attackers, and absence of key confirmation. Similarly, the study in [3] presents another thorough formal analysis, considering all four essential components of a real-world 5G-AKA protocol and various channel-compromised scenarios. This analysis, also conducted using Tamarin Prover [24], reveals that the security of the 5G-AKA protocol relies on underspecified assumptions regarding the inner workings of underlying channels, potentially leading to security-critical race conditions. Furthermore, a recent formal analysis of 5G-AKA is outlined in [4], focusing specifically on the different phases within the authenticated key agreement procedure and their impact on critical mobile-network objects such as Protocol Data Unit (PDU) sessions. In addition to the formal analysis of the 5G-AKA protocol, its predecessors, such as 3G/4G, have also undergone extensive scrutiny in [5], [7], and [25].

III. 5G-AKA Protocol

In this section, we present an overview of the 5G-AKA protocol as defined by the 3GPP TS 33.501 standard [1], followed by a brief discussion of its associated privacy issues in Section III-A1. This overview aims to highlight the distinctions between 5G-AKA and our proposed protocols, as detailed in the subsequent sections of this paper. Additionally, it emphasizes the need for a more secure AKA protocol within the 5G framework. Following the examples of [2] and [13], we did not consider all components of the HN for simplicity.

The 5G-AKA protocol uses several cryptographic primitives such as hash functions $f1, f2, f3, f4, f5, f1^*, f5^*$, the Secure Hash Algorithm-256 (SHA256), the Elliptic Curve Integrated Encryption Scheme (ECIES) for key encapsulation (please refer to Appendix A for more details), and the Key Derivation Function (KDF). More details on the cryptographic primitives can be found in 3GPP TS 33.501 [1].

As mentioned earlier, the 5G-AKA protocol involves three primary entities: UE, SN, and HN. The subscriber or UE represents the mobile device user with the USIM card and connects to the 5G network to access its services. The USIM card typically contains essential information, including the Subscription Permanent Identifier (SUPI), the long-term secret key (k), the sequence number ($\mathsf{SQN_{UE}}$), and the HN's public key ($\mathsf{PK_{HN}}$). The SN is the network carrier that the subscriber connects to, while the HN is the subscriber's own network carrier. The main functionality of the 5G-AKA protocol is to provide mutual authentication and establish a session key between the entities. A high-level overview of the 5G-AKA protocol is shown in Figure 1. We provide a detailed overview of the 5G-AKA protocol in Appendix B. Here, we briefly touch upon each phase. 5G-AKA consists of four phases: *Initiation* (Phase 1), *Challenge-Response* (Phase 2), *Sequence Number Re-synchronization* (Phase 3), and *MAC Failure* (Phase 4).

Initiation (Phase 1): In the Initiation phase, the subscriber (i.e., the UE) sends the encrypted SUPI using ECIES, which is represented as SUCI (Subscription Concealed Identifier), to the HN via the radio channel through an SN.

Challenge-Response (Phase 2): In the Challenge-Response phase, the HN chooses a random value R. It computes AUTN, which consists of the MAC (Message Authentication Code) for the R and the XOR ($\oplus$) of the sequence number $\mathsf{SQN_{HN}}$ for privacy. The subscriber performs two tasks upon receiving the challenge message from the HN. First, it checks the MAC to authenticate and verify the integrity of the challenge R. Second, the $\mathsf{SQN_{HN}}$ is used to check the freshness of the challenge R from the HN to prevent replay attacks. The subscriber checks the freshness of the challenge message by comparing the received $\mathsf{SQN_{HN}}$ with its own

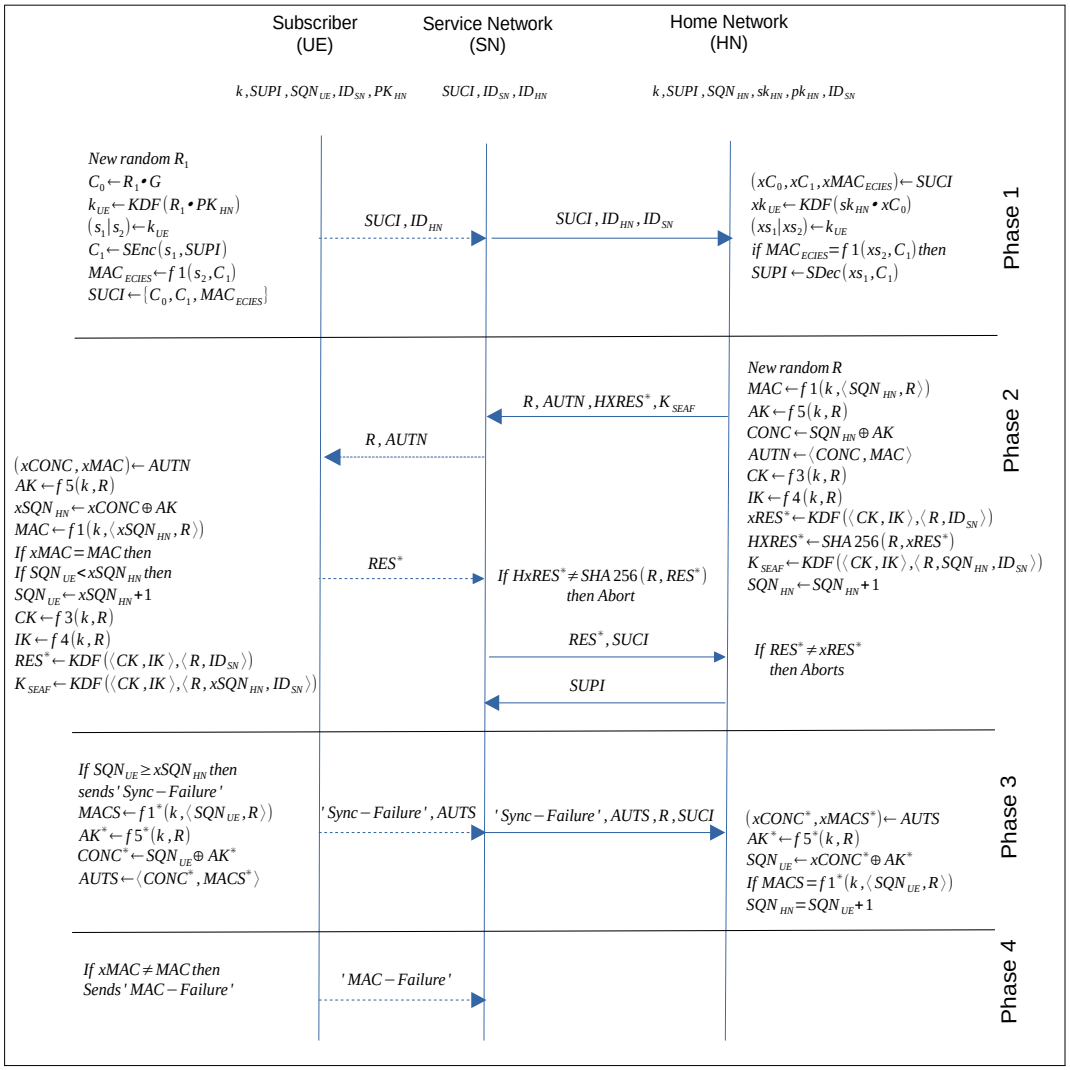

Fig. 1: A high-level overview of the 5G-AKA protocol, where dotted and solid arrows represent open channels and authenticated secure channels, respectively.

sequence number $\mathsf{SQN_{UE}}$. Please note that, as per the 3GPP TS 33.50 [1], $\mathsf{SQN_{HN}}$ and $\mathsf{SQN_{UE}}$ should remain synchronized at all times.

Sequence Number Re-synchronization (Phase 3): If this comparison of the sequence numbers fails, i.e., if $\mathsf{SQN_{HN}}$ and $\mathsf{SQN_{UE}}$ are unsynchronized, the subscriber sends a SYNC_FAILURE message, along with AUTN, to the HN for re-synchronization of its sequence number $\mathsf{SQN_{UE}}$ with the HN's $\mathsf{SQN_{HN}}$. Afterward, the subscriber returns to the Initiation phase to restart the process from the beginning.

MAC Failure (Phase 4): If the MAC check fails, the subscriber outputs a MAC_FAILURE message and returns to the Initiation phase to restart the process. If both the MAC and $\mathsf{SQN_{HN}}$ checks are successfully completed, the subscriber generates a response message RES for the challenge R, computes the key material to generate the anchor keys (i.e., $\mathsf{K_{SEAF}}$), and sends RES to the HN.

A. Analysis of 5G-AKA Protocol Limitations

In this section, we briefly revisit the key challenges of the 5G-AKA protocol introduced in Section I.

1) Privacy Threats in 5G-AKA from Active Attackers: The 5G-AKA protocol is primarily vulnerable to three types of privacy threats that compromise subscriber privacy. These threats aim to violate the property of unlinkability, which safeguards genuine subscribers from being uniquely identified or distinguished. Below, we summarize the three linkability attacks targeting subscribers, with detailed explanations provided in Appendix C.

Failure Message Linkability Attack [5]: This attack aims to distinguish a targeted subscriber from others by replaying

records of $\langle R, \mathrm{AUTN} \rangle$ to all subscribers in the vicinity and analyzing their response patterns.

Sequence Number Inference Attack [7]: The objective of this attack is to infer information about the targeted subscriber's sequence number $\mathrm{SQN}_{\mathrm{UE}}$ by repeatedly replaying previously captured $\langle R, \mathrm{AUTN} \rangle$ tuples.

Encrypted SUPI Replay Attack [8], [9]: This attack seeks to identify the targeted subscriber by replaying a captured SUCI during the Initiation phase across all subscriber sessions, then analyzing the responses to the corresponding challenge messages from the Home Network (HN).

2) Lack of PFS: As shown in Figure 1, the anchor key $\mathsf{K_{SEAF}}$ is derived from the cipher key CK, integrity key IK, the random challenge R, the sequence number $\mathsf{SQN_{HN}}$, and the serving network identity $\mathsf{ID_{SN}}$. The keys CK and IK are generated by the home network based on the subscriber's long-term secret key k and the random number R, where R is sent over the open channel to the subscriber. Additionally, the sequence number $\mathsf{SQN_{HN}}$ can be recovered from the concealed value CONC by first deriving the anonymity key AK using the long-term key k and R. Since both R and CONC are transmitted in the clear, an attacker with access to the long-term key k can recover $\mathsf{SQN_{HN}}$. This implies that if the long-term key k is ever compromised and the attacker has recorded previous protocol runs, they can retroactively compute CK, IK, and eventually $\mathsf{K_{SEAF}}$. As a result, both past and future session keys are at risk, meaning the 5G-AKA protocol does not offer PFS.

3) Inefficient SQN-based Replay Attack Prevention: As shown in Phase 2 of Figure 1, the subscriber compares its stored sequence number $\mathsf{SQN_{UE}}$ with the sequence number $\mathsf{xSQN_{HN}}$ received from the HN. For authentication to succeed, these two values must match. Both the subscriber and the HN increment their respective sequence numbers during the authentication process. However, if the sequence numbers become misaligned- due to network issues, message loss, or delays- the authentication will fail. This failure triggers Phase 3, the sequence number (SQN) resynchronization process, which aims to restore synchronization between the two parties. This additional step increases communication overhead and adds complexity to the protocol. Furthermore, as discussed in Section III-A1, the use of sequence numbers can introduce privacy risks. In particular, it opens the door to attacks such as the Sequence Number Inference Attack [7], where an attacker may deduce information about a subscriber's activity based on sequence number behavior.

IV. System Model, Threat Model & Security Requirements

In this section, we present the system model, threat model, and security requirements for our protocols, which are primarily based on the 3GPP standards (3GPP TS 33.501 [1]).

A. System Model

Our protocols consider three broad roles in the system model similar to Basin *et al.* [2]: *Subscriber* or UE, HN, and SN. Figure 2 shows the entities/roles and channels involved in our protocols. The subscribers (UEs) represent smartphones or IoT devices equipped with USIMs. The USIM is a cryptographic chip that stores subscriber-related information, such as the long-term secret key (k) and the unique identity of the subscriber, known as "Subscription Permanent Identifier" or SUPI. Each subscriber is registered with an HN. The HN is typically the network carrier of the subscriber, which maintains the database of the subscribers and also performs authentication before providing access to its services. It also stores the same subscriber-related information as the USIM for each registered subscriber. In the real world, the HN consists of multiple sub-entities, such as the Authentication Server Function (AUSF), which authenticates subscribers, and Unified Data Management (UDM), which computes keying material and authentication-related data for the AUSF. The UDM consists of the Authentication Credential Repository and Processing Function (ARPF) and Subscription Identifier De-concealing Function (SIDF). The primary purpose of ARPF is to store subscribers' secret credentials and compute authentication-related cryptographic parameters. On the other hand, SIDF contains the HN's private key, which recovers the plaintext SUPI of subscribers from its encrypted version SUCI. For simplicity and without compromising overall security, we consider AUSF and UDM as a single entity, which is HN. The SNs are the actual network carriers to which subscribers connect and get access to the cellular network. When subscribers are within their HN coverage area, the HN also serves as the SN. In roaming scenarios, however, the SN is operated by a different network provider. We note that, similar to the 5G-AKA protocol, our proposed protocols are applicable in both roaming and non-roaming scenarios.

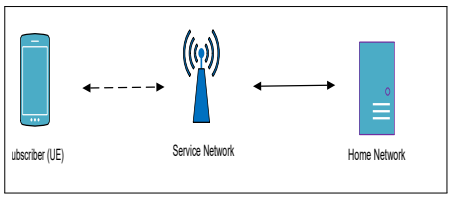

Fig. 2: Overall Architecture, where dotted and solid arrows represent open channels and authenticated secure channels, respectively.

Remark 1: In our model, we abstract the UE and USIM as a single logical entity, consistent with previous works (e.g., [2], [13]) and common in formal protocol analysis. However, we acknowledge that in practical deployments, SUCI (i.e., the encrypted SUPI) may be generated either on the USIM or on the host UE, depending on the SIM profile and configuration, as specified in 3GPP TS 31.121 [26] and TS 33.501 [1]. When SUCI is generated on the UE, the SUPI must traverse the SIM–UE interface in plaintext, which introduces a potential attack surface. Recent works such as SecureSIM [27] and SIMurai [28] have demonstrated that this interface can be vulnerable to runtime abuse, man-in-the-middle attacks, and

software-level compromise.

Our analysis assumes that SUCI is generated within the USIM, which is the recommended and more secure configuration according to 3GPP specifications. We highlight this assumption to clarify the trust boundary of the subscriber entity in Protocols I and II, presented in the following section. In cases where SUCI is instead computed on the UE, we assume the presence of existing protection mechanisms, such as multi-factor access control and runtime isolation, as proposed in SecureSIM [27] and SIMurai [28].

B. Threat Model & Security Assumptions

Our threat model incorporates all the requirements outlined in the 3GPP standards (TS 33.501 [1]). Additionally, our protocols take into account the additional security requirements highlighted in [2] and [3].

Assumption on Channels: In accordance with 3GPP standards, the communication channel between SNs and HNs is considered authenticated and secure (i.e., a private channel). This ensures that any attempt by an attacker to eavesdrop, insert, or modify messages within the channel will be prevented and detected. However, the communication channel between the subscriber and the SN is considered insecure or open, allowing passive attackers to eavesdrop and active attackers to manipulate, intercept, and inject messages. Research by Basin et al. [2] and Cremers et al. [3] emphasizes the necessity for a binding channel between SNs and HNs, where each message is associated with a unique session ID to maintain security. Our protocols operate under the assumption of such channel binding between SNs and HNs.

Assumption on Cryptographic Functions: Our protocols do not require all the cryptographic primitives outlined in 3GPP TS 33.105 [29]. Instead, it utilizes a part of the cryptographic functions such as $\mathsf{SHA256}$, $\mathsf{f1}$, $\mathsf{f2}$, $\mathsf{f3}$, and $\mathsf{f4}$. We assume that these cryptographic functions are publicly available and provide confidentiality and integrity of their inputs. It is important to note that our protocols do not utilize the cryptographic functions $\mathsf{f5}$, $\mathsf{f1}^*$, and $\mathsf{f5}^*$ from the 5G-AKA protocol. Additionally, the Elliptic Curve Integrated Encryption Scheme (ECIES) [30] is employed as a secure public-key encryption. These assumptions regarding cryptographic functions are aligned with the requirements outlined in the 3GPP standards.

Assumption on Components: Aligned with 3GPP standards, our threat model assumes the possibility of certain SNs and HNs being compromised. The long-term secret key (k) of honest subscribers always remains secure, and the honest subscribers are capable of protecting their anchor keys ($\mathsf{K_{SEAF}}$). However, in proving our PFS property, we consider a scenario where the attacker has access to both the long-term secret key (k) of the subscribers and the private key ($\mathsf{sk_{HN}}$) of the HN. Please note that the long-term secret key $\mathtt{k}$ of the subscriber may occasionally need to be updated for security reasons. In such cases, the mobile network operator can employ an over-the-air (OTA) SIM provisioning process to update the subscribers' long-term secret keys. However, such mechanisms fall outside the scope of our protocol design.

C. Security Requirements

The security requirements outlined in the 3GPP security specifications can be broadly categorized into three groups: privacy, secrecy, and authentication [13]. We define all the necessary security requirements specified in the 3GPP security specifications, along with the additional security requirements outlined in [2] and [3], which are not fully specified by 3GPP.

Privacy: According to the 3GPP TS 33.501 [1] standards, a subscriber's privacy requirements can be classified into three types: user identity confidentiality, user location confidentiality, and user untraceability. These privacy requirements can be met by safeguarding the secrecy of the SUPI and ensuring the subscriber's untraceability, as demonstrated in [2][4]. Our protocols should support these privacy requirements in the presence of both passive and active attackers, as indicated in [2]. Furthermore, Wang *et al.* [13] demonstrated that all mentioned privacy requirements can be achieved if our protocol supports the indistinguishability property. This property asserts that no attacker can distinguish between two subscribers. We summarize this property below:

Subscriber Indistinguishability: If there are two subscribers, denoted as UE1 and UE2, and an AKA session involves UE1 (or UE2), no active attacker can distinguish whether it is engaged with UE1 or UE2.

Secrecy: In the 3GPP TS 33.501 [1], in addition to the secrecy of the SUPI and long-term secret key (k), there is also the requirement for the secrecy of the anchor keys ($\mathsf{K_{SEAF}}$). As demonstrated in [2], the 5G-AKA protocol does not provide PFS. Therefore, our protocol II aims to support PFS as well. We outline the two secrecy requirements below:

Key Secrecy: The anchor key ($\mathsf{K_{SEAF}}$) must remain secret.

Perfect Forward Secrecy: If the long-term secret key (k) and the long-term private key ($\mathsf{sk_{HN}}$) of the HN are compromised, the attacker must not be able to compute any previously generated anchor keys ($\mathsf{K_{SEAF}}$).

Authentication: Our protocols also aim to provide all the authentication requirements specified in [2] and [3], which are formulated from the 3GPP TS 33.501 [1] standards. In [2] and [3], the authors used Lowe's taxonomy [31] to represent the authentication properties. We define the following definitions based on Lowe's taxonomy [31]:

- *Weak Agreement*: Weak agreement occurs when participant A (acting as the initiator) completes a protocol run with participant B, and participant B must have previously participated in the protocol with participant A.
- *Non-Injective Agreement*: Non-injective agreement means that participant A completes a protocol run with participant B, and participant B must have previously partici-

[4]Please note that our protocols do not require sequence numbers, unlike 5G-AKA. Therefore, the privacy requirement of the subscriber in our protocols is independent of the sequence number.

pated in the protocol with participant A, with both A and B agreeing on the contents of all the messages exchanged.
- *Injective Agreement*: Injective agreement is a stronger version of non-injective agreement, requiring a one-to-one correspondence between the runs of A and B. It ensures that each run of A corresponds to a unique run of B, with both participants agreeing on the data or secrets involved.

We list the required authentication properties of our protocols below:

Agreement between UE and SN: The subscriber and SN must both obtain injective agreement on $\mathsf{K_{SEAF}}$ and weak agreement with each other.

Agreement between UE and HN: The subscriber and HN must achieve injective agreement regarding $\mathsf{K_{SEAF}}$ and establish a weak agreement with each other. Additionally, both the subscriber and the HN must achieve a non-injective agreement regarding $\mathsf{ID_{SN}}$ and SUPI with each other.

Agreement between SN and HN: The SN and HN must both achieve injective agreement on $\mathsf{K_{SEAF}}$ and weak agreement with each other. The SN must also achieve a non-injective agreement on SUPI with HN.

V. Our Proposed Protocol

Our protocol aims to streamline the AKA mechanism in 5G by addressing the limitations of the existing 5G-AKA protocol, such as enhancing subscriber privacy, ensuring PFS, and introducing an efficient replay attack prevention method. The objective is to meet all the security requirements outlined in the 3GPP standards and accommodate additional crucial security needs that are not fully specified in the standards, as highlighted in [2]. This section begins with an overview of Protocol I. Then, we provide a detailed description of the protocol in Section V-B. In Section V-C, we present Protocol II, an extension of Protocol I to support PFS.

A. Overview

As pointed out in [2], the 5G-AKA protocol exhibits vulnerabilities against active attackers. Several attack scenarios aimed at compromising subscriber privacy have been illustrated [2], [9], [5], [7] particularly focusing on capturing and replaying messages from targeted subscribers. The idea used by the attacks is to distinguish the responses elicited from the targeted subscriber and other subscribers to the same replayed messages. This enables attackers to differentiate them, thus breaching the privacy of the targeted subscriber. Appendix C presents a brief overview of the common types of privacy-related attacks in the 5G-AKA protocol and the methods used to carry out these attacks.

The primary vulnerability in 5G-AKA lies in conducting two security checks at the subscriber side for the HN's challenge message, as highlighted in [13]. Initially, the subscriber verifies the authenticity and integrity of the HN's random challenge by checking the MAC. Subsequently, the subscriber assesses the freshness of the HN's challenge by comparing the subscriber's sequence number $\mathsf{SQN_{UE}}$ with the HN's sequence number $\mathsf{SQN_{HN}}$. For instance, in the *Encrypted SUPI Replay Attack* [9], [5], the attacker captures the SUCI of the targeted subscriber and replays it later. Consequently, the targeted subscriber responds to the HN's challenge, whereas other subscribers return MAC_FAILURE responses. Similarly, in the *Failure Message Linkability Attack* [5], [2], the attacker captures R and AUTN sent by the HN to the targeted subscriber and replays them later. As a result, the targeted subscriber successfully verifies the MAC but fails to confirm freshness, responding with a SYNC_FAILURE message. Conversely, all other subscribers fail the MAC verification and respond with MAC_FAILURE messages. The authors of [7] demonstrated another type of privacy-related attack called the *Sequence Number Inference Attack* in the 5G-AKA protocol by capturing R and AUTN and replaying them later.

We observe two primary reasons in the 5G-AKA protocol for its lack of resistance to active attackers. Firstly, there is a lack of binding between the subscriber's initial authentication request (i.e., SUCI) and the corresponding HN's challenge (R, AUTN). The subscriber cannot ascertain whether the HN's challenge is linked to its most recent authentication request. Secondly, the freshness check (or prevention of replay attacks) relies on a sequence number maintained by both the subscriber and HN.

Our protocol I is designed to address the above-mentioned issues in the 5G-AKA protocol while ensuring that all essential security requirements outlined in Section IV are met. Unlike the 5G-AKA protocol, which relies solely on the HN sending the challenge and the subscriber's corresponding response, our protocol I incorporates an additional challenge message from the subscriber to the HN during the initial authentication request. These challenge and response messages are mutually bound using random numbers, eliminating the need for a sequence number-based replay attack prevention mechanism for freshness checking. This, in turn, also reduces the complexity of our protocol compared to the 5G-AKA protocol. Next, we provide a detailed description of Protocol I.

B. Main Construction

Figure 3 provides a high-level overview of our protocol I. It is divided into three phases: *Initiation & UE's Challenge*, *HN's Challenge & Response* and *UE's Response & Key Confirmation*, as will be described shortly. Our protocol I uses the cryptographic functions such as, $f1, f2, f3, f4$, KDF and SHA256, as documented in 3GPP TS 33.501 [1].

Initiation & UE's Challenge The subscriber (UE) initiates this phase when the SN triggers authentication. In this phase, the subscriber primarily sends two important messages to the HN: the encrypted SUPI, which is SUCI, and a challenge R. The subscriber mainly performs two tasks, as outlined below.

Firstly, the subscriber chooses a random challenge R and generates a randomized encrypted SUPI for privacy protection using ECIES, as outlined in Appendix A. The ECIES key encapsulation mechanism $\mathsf{Encap_{ECIES}}$ generates a shared secret key $\mathsf{k_{UE}}$ and an ephemeral public-key component C_0. Concurrently, the data encryption mechanism of ECIES produces

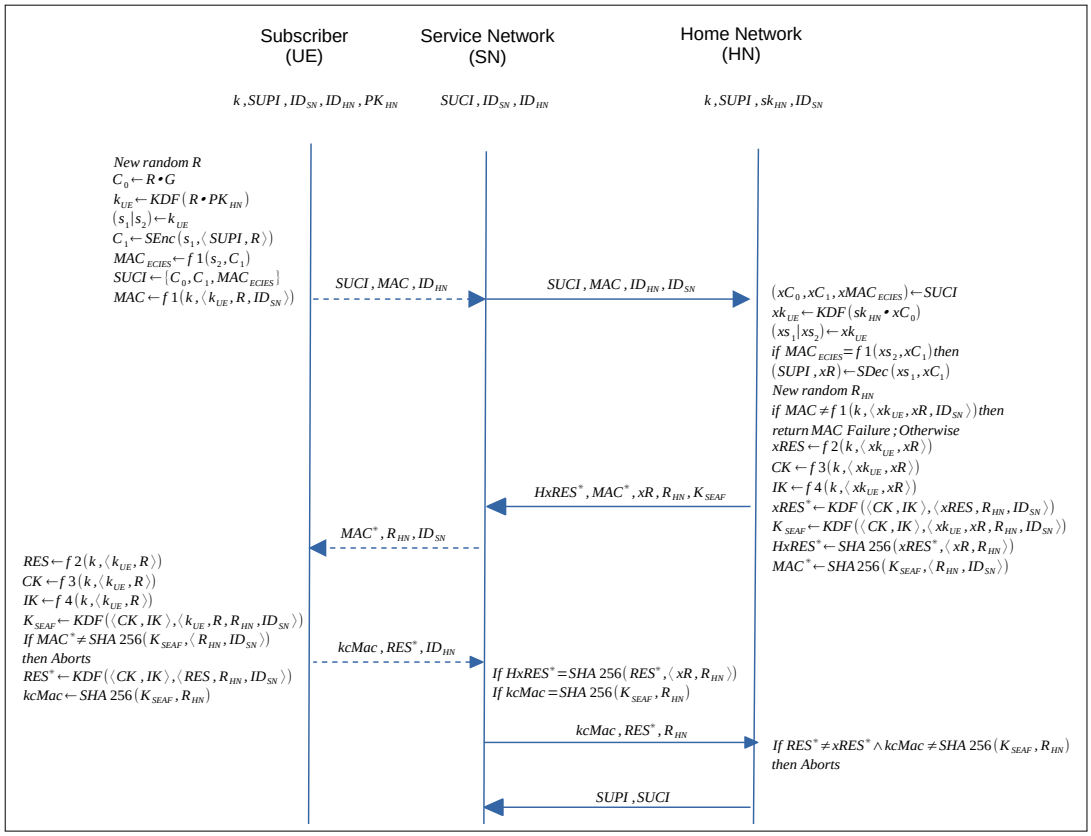

Fig. 3: A high-level overview of our protocol I, where dotted and solid arrows represent open channels and authenticated secure channels, respectively.

an encrypted component C_1 for the SUPI and the random challenge R using symmetric key encryption. The $\mathsf{s_1}$ portion of the shared secret key $\mathsf{k_{UE}}$ is used as the encryption key, while the remaining $\mathsf{s_2}$ portion creates a tag (i.e., message authentication code) $\mathsf{MAC_{ECIES}}$ for authenticity and integrity verification of C_1.

Secondly, a message authentication code MAC is computed for the random challenge R. This will verify the authenticity and integrity of the challenge R.

Finally, the subscriber sends the tuple $\text{SUCI} = \langle C_0, C_1, \text{MAC}_{\text{ECIES}} \rangle, \text{MAC}, \text{ID}_{\text{HN}}$ to the SN. Then, the SN forwards the tuple $\langle \text{SUCI}, \text{MAC}, \text{ID}_{\text{HN}}, \text{ID}_{\text{SN}} \rangle$ to the HN.

HN's Challenge & Response In this phase, the HN performs broadly two tasks once the HN receives the tuple $\langle \text{SUCI}, \text{MAC}, \text{ID}_{\text{HN}}, \text{ID}_{\text{SN}} \rangle$ from a subscriber.

Firstly, the HN recovers the plaintext SUPI using its private key $\mathsf{sk_{HN}}$ from SUCI to retrieve the long-term secret key k associated with the subscriber from its database. Following the decapsulation operation of ECIES using its private key $\mathsf{sk_{HN}}$ and the ephemeral public-key component C_0 of SUCI, the HN acquires the shared secret key $\mathsf{xk_{ue}}$. Subsequently, it decrypts C_1 using $\mathsf{xs_1}$ (a component of the shared secret $\mathsf{xk_{ue}}$) after successfully verifying the tag (i.e., $\mathsf{MAC_{ECIES}}$) using $\mathsf{xs_2}$ (the second component of the shared secret $\mathsf{xk_{ue}}$) and C_1, obtaining the plaintext SUPI and the subscriber's challenge xR.

Secondly, the HN verifies the authenticity and integrity of the subscriber's challenge xR by checking the MAC. An unsuccessful verification results in a MAC_FAILURE. Once the verification is successful, the HN chooses a random challenge for the subscriber R_{HN} and computes xRES, cipher key CK, and integrity key IK using the long-term secret key k of the subscriber and the challenge xR. It then generates the expected response from the subscriber xRES^*, and the anchor keys $\mathsf{K_{SEAF}}$. Next, the HN computes the hashed response HxRES^*, which includes the expected response xRES^*, the subscriber's challenge xR, and its challenge for the subscriber R_{HN}. Additionally, the HN generates a message authentication code MAC^*. The HN sends the tuple $\langle \text{HxRES}^*, \text{MAC}^*, xR, R_{HN}, \mathsf{K_{SEAF}} \rangle$ to the SN. Note that MAC^* serves three important functions: it acts as the response from the HN to the subscriber's challenge, it provides authenticity and integrity verification for the HN's challenge R_{HN}, and it offers key confirmation for the anchor key $\mathsf{K_{SEAF}}$ to the subscriber.

Upon receiving the tuple $\langle \text{HxRES}^*, \text{MAC}^*, xR, R_{HN}, \mathsf{K_{SEAF}} \rangle$, the SN retains $\langle \text{HxRES}^*, xR, R_{HN}, \mathsf{K_{SEAF}} \rangle$ and forwards the tuple $\langle \text{MAC}^*, R_{HN}, \text{ID}_{\text{SN}} \rangle$ to the subscriber.

UE's Response & Key Confirmation: Upon receiving the tuple $\langle \mathrm{MAC}^*, R_{HN}, \mathrm{ID_{SN}} \rangle$, the subscriber computes the response RES, cipher key CK, and integrity key IK using $f2$, $f3$, and $f4$ respectively, along with the long-term secret key k, shared secret key $\mathsf{k_{UE}}$, and the random challenge R as inputs. The subscriber then computes the anchor keys $\mathsf{K_{SEAF}}$ and verifies the MAC^* using $\mathsf{K_{SEAF}}$, HN's challenge R_{HN}, and $\mathrm{ID_{SN}}$.

If the verification fails, the connection is aborted. If the verification is successful, it provides three functions: authenticity and integrity of the HN's challenge R_{HN}, confirmation of the anchor key $\mathsf{K_{SEAF}}$, and verification of the HN's response to its challenge R. Afterward, the subscriber generates its response RES^* and a message authentication code kcMAC using SHA256, and sends the tuple $\langle \mathrm{kcMAC}, \mathrm{RES}^*, \mathrm{ID_{SN}} \rangle$ to the SN. Note that kcMAC serves two purposes: verifying the subscriber's response to the HN's challenge R_{HN} and confirming the anchor keys $\mathsf{K_{SEAF}}$ to the HN.

Upon receiving the response from the subscriber, SN first verifies HxRES with RES^*, xR and R_{HN}. It also verifies kcMAC with the anchor keys $\mathsf{K_{SEAF}}$ and R_{HN}. A successful verification indicates authentication of the subscriber at the SN and anchor key $\mathsf{K_{SEAF}}$ confirmation. Next, the SN forwards the tuple $\langle \mathsf{kcMAC}, \mathrm{RES}^*, R_{HN} \rangle$ to the HN, where R_{HN} is sent for the binding purposes.

Upon receiving the tuple $\langle \mathsf{kcMAC}, \mathrm{RES}^*, R_{HN} \rangle$, the HN verifies RES^* with xRES^* and also kcMAC using $\mathsf{K_{SEAF}}$ and R_{HN}. A successful verification indicates mutual authentication between the subscriber and the HN and confirms the anchor keys $\mathsf{K_{SEAF}}$. The HN then sends the tuple $\langle \mathrm{SUPI}, \mathrm{SUCI} \rangle$ to the SN, where SUCI is used for binding purposes. Once the SN receives SUPI, the authentication process is concluded, and the subscriber can use the anchor key $\mathsf{K_{SEAF}}$ to access network services.

C. Extension for Perfect Forward Secrecy (PFS)

In this section, we present Protocol II, an extension of Protocol I, designed to support PFS with minimal additional computational overhead. Please note that Protocol I is a foundational enhancement to the 5G-AKA protocol, addressing key limitations without relying on PFS. It replaces sequence number-based replay protection with a stateless challenge-response mechanism, simplifying synchronization and reducing overhead. This design also improves resistance to active attacks by binding authentication messages between the subscriber and the home network. Crucially, Protocol I remains compatible with existing systems, making it practical for deployment in current 5G networks.

Protocol II builds on this foundation by introducing PFS, which protects past session keys even if long-term secrets are later compromised, which is a growing concern in the face of persistent and sophisticated adversaries. This layered design ensures that Protocol I is practical and deployable on its own, while Protocol II offers additional security for more demanding threat models.

To achieve PFS, our previously described Protocol I requires some modifications. Figure 4 shows our modified protocol, with the highlighted portions indicating the new changes. The core addition in our modified protocol II is the introduction of an ephemeral DH key exchange between the subscriber and the HN. Please note that our modified protocol II supports all the security requirements of our protocol I, which are confirmed by our formal analysis.

As we aim to maintain USIM compatibility without introducing new cryptographic operations, our core approach is to utilize the ECIES ephemeral public key C_0 as DH key material for computing a shared DH key between the subscriber and the HN. In Figure 4, it is shown that the HN employs the ephemeral public key xC_0 (i.e., C_0) to calculate a DH key $\mathsf{dh_{key}}$, using its random challenge R_{HN} as the secret key. Furthermore, the HN computes the DH key material dh_{HN} for the subscriber using R_{HN}, enabling the subscriber to compute the same DH key $\mathsf{dh_{key}}$ on its end. Subsequently, both the subscriber and the HN derive the anchor keys (i.e., $\mathsf{K_{SEAF}}$) using the $\mathsf{dh_{key}}$, $\mathsf{k_{UE}}$, and $\mathrm{ID_{SN}}$. To avoid redundancy, we refrain from reiterating the entire protocol, as the remaining concept aligns with our protocol I.

TABLE II: Security Requirements Achieved by Our Protocols

Point of View	UE		SN		HN	
Partner	SN	HN	UE	HN	UE	SN
Weak agreement	✓	✓	✓	✓	✓	✓
Agreement on $\mathsf{K_{SEAF}}$	I	I	I	I	I	I
Agreement on $\mathsf{ID_{SN}}$	NI	NI	NI	NI	NI	NI
Agreement on SUPI	NI	NI	NI	NI	NI	NI
Secrecy on $\mathsf{K_{SEAF}}$	✓		✓		✓	
Secrecy on SUPI	✓		✓		✓	
PFS*	✓					
Indistinguishability	✓					

I: Injective agreement; wa: Weak agreement; NI: Non-injective agreement; ✓: Property supported; *: only our protocol II support PFS.

VI. Formal Security Analysis

ProVerif is a state-of-the-art symbolic model-based automated formal analysis tool that uses applied π-calculus syntax [32]. It operates under the *Dolev-Yao Attack Model* [33], which grants the attacker the ability to read, modify, delete, and forge packets, as well as inject them into the public communication channel. ProVerif evaluates whether the designed protocol meets the defined security objectives within this attack model. When an attack is detected, ProVerif provides a comprehensive description of the steps involved in the attack. ProVerif is ideal for stateless protocols like ours and for verifying a wide range of security properties under the Dolev-Yao model, including indistinguishability properties[5]. For our analysis, we use ProVerif version 2.05 [32]. Our modeling codes are available in https://anonymous.4open.science/r/AKA-5G-ProVerif-9378/.

[5]5G-AKA is a stateful protocol due to its sequence number-based replay attack prevention mechanism, which makes it unsuitable for modeling using ProVerif. Consequently, most existing formal analyses of the 5G-AKA protocol are based on Tamarin [24], which supports stateful protocols.

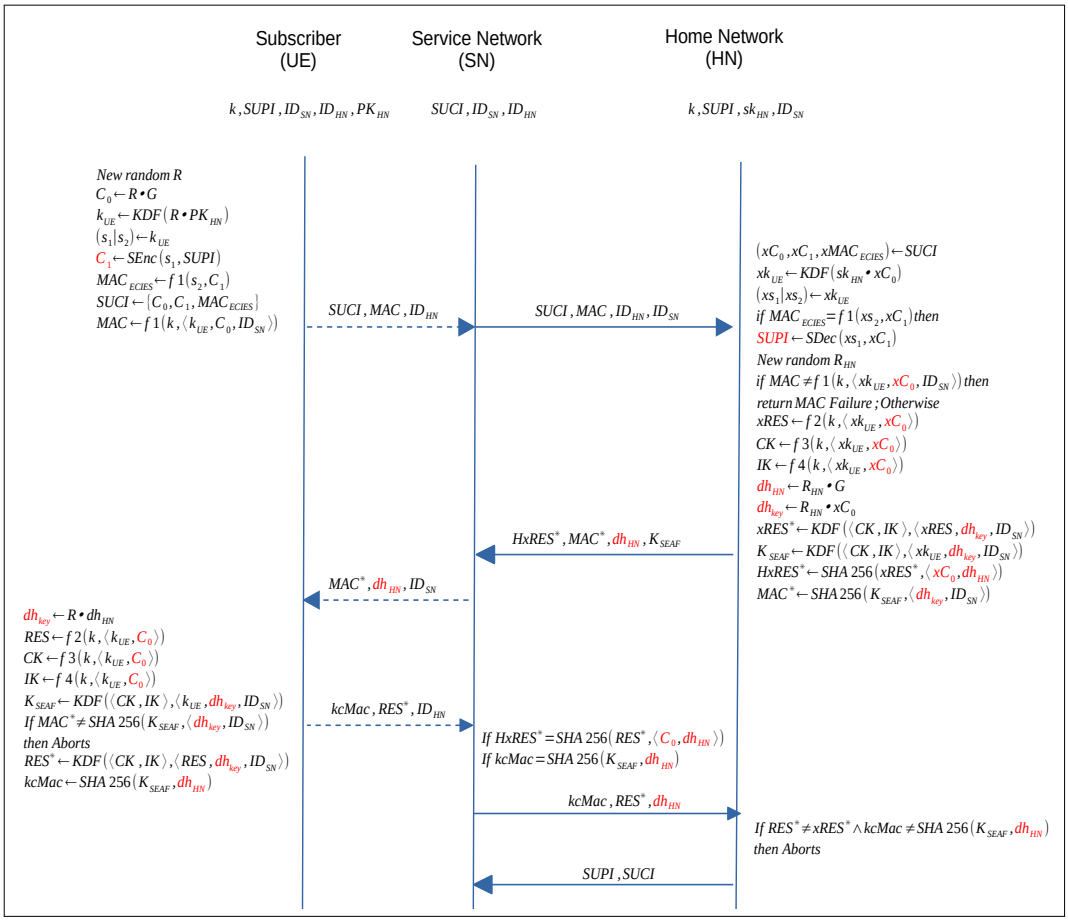

Fig. 4: A high-level overview of our protocol II, where dotted and solid arrows represent open channels and authenticated secure channels, respectively.

		5G-AKA [1]	5G-AKA′ [13]	Our Protocol I	Our Protocol II
Resynchronization		Yes	Yes	No	No
Messages	Resync	13	13	NA	NA
	No Resync	9*	9*	7	7

*5G-AKA and 5G-AKA′ require 7 message interactions between the parties for mutual authentication and an additional 2 messages for the anchor key K_{SEAF} confirmation between the subscriber and the SN; NA: Not Applicable.

TABLE III: Communication overhead comparison between our protocols (Protocols I and II) and 5G-AKA [1] and 5G-AKA′

	Case 1			Case 2			Case 3		
	UE	SN	HN	UE	SN	HN	UE	SN	HN
5G-AKA [1]	$8T_h + 2T_m + T_{enc} + T_{xor} + T_{add} + T_{prf}$	T_h	$9T_h + T_m + T_{dec} + T_{xor} + T_{add} + T_{prf}$	$6T_h + 2T_m + T_{enc} + 2T_{xor} + T_{prf}$	T_h	$11T_h + T_m + T_{dec} + 2T_{xor} + 2T_{add} + T_{prf}$	$4T_h + 2T_m + T_{enc} + T_{xor} + T_{prf}$	T_h	$9T_h + T_m + T_{dec} + T_{xor} + T_{add} + T_{prf}$
5G-AKA′ [13]	$8T_h + 2T_m + T_{enc} + T_{xor} + T_{add} + T_{prf} + T_{dec}$	T_h	$9T_h + T_m + T_{enc} + T_{xor} + T_{add} + T_{prf} + T_{dec}$	$6T_h + 2T_m + T_{enc} + 2T_{xor} + T_{prf} + T_{dec}$	T_h	$11T_h + T_m + T_{enc} + 2T_{xor} + 2T_{add} + T_{prf} + T_{dec}$	$4T_h + 2T_m + T_{enc} + T_{xor} + T_{prf} + T_{dec}$	T_h	$9T_h + T_m + T_{enc} + T_{xor} + T_{add} + T_{prf} + T_{dec}$
Our Protocol I	$10T_h + 2T_m + T_{enc} + T_{prf}$	$2T_h$	$11T_h + T_m + T_{dec} + T_{prf}$	NA	NA	NA	$8T_h + 2T_m + T_{enc} + T_{prf}$	$2T_h$	$10T_h + T_m + T_{dec} + T_{prf}$
Our Protocol II	$10T_h + 3T_m + T_{enc} + T_{prf}$	$2T_h$	$11T_h + 3T_m + T_{dec} + T_{prf}$	NA	NA	NA	$8T_h + 3T_m + T_{enc} + T_{prf}$	$2T_h$	$10T_h + 3T_m + T_{dec} + T_{prf}$

TABLE IV: Theoretical Computation Cost Comparison between Our Protocol I, Protocol II, 5G-AKA [1] and 5G-AKA′ [13]

A. *Modeling Choices*

In formal verification, modeling choices play a crucial role in defining the behavior, assumptions, and properties of the system to facilitate formal analysis. These choices abstractly represent the system or protocol under analysis. In this section, we will briefly outline the modeling choices made for our

	Case 1			Case 2			Case 3		
	UE	SN	HN	UE	SN	HN	UE	SN	HN
5G-AKA [1]	39609.97	22.57	38529.26	31340.20	21.14	38614.81	18600.93	21.42	39314.69
5G-AKA′ [13]	40539.53	23.49	39572.49	32023.00	24.29	39820.31	19107.45	23.05	39450.83
Our Protocol I	39631.14	32.71	40627.80	NA	NA	NA	28835.33	32.28	40436.40
Our Protocol II	39664.36	30.96	46579.51	NA	NA	NA	28857.77	31.73	46062.19

TABLE V: Computation Time (in microseconds) Comparison between Our Protocol I, Protocol II, 5G-AKA [1] and 5G-AKA′ [13]

protocols to provide a better understanding of our modeled system.

Architecture: We consider three roles: subscribers, SNs, and HNs. Each role can have an unbounded number of instances. The communication channels between the subscriber and SN, and between the SN and HN, are considered open (or insecure) and authenticated private (or secure) channels, respectively, as mentioned in Section IV.

Modeling Cryptographic Primitives: We model symmetric key-based encryption and decryption operations using the constructor senc for encryption and the destructor sdec for decryption. The senc constructor takes two arguments: a message m of type bitstring and a key n of type bitstring, and returns the encrypted message. The sdec destructor takes an encrypted message produced by senc and the corresponding key n, and returns the original message m. Additionally, cryptographic hash functions such as $f1$, $f2$, $f3$, $f4$, KDF, and SHA256 are also modeled as constructors. We model the modular exponentiation operation (i.e., g^x) using the constructor exp. This constructor takes two arguments of type G and exponent. For modeling the DH Key Exchange, we utilize the *equation* concept in ProVerif. The DH Key Exchange is represented by the equation:

$$\begin{aligned}&\text{const g : G [data].}\\&\text{fun exp(G, exponent) : G.}\\&\text{equation forall x : exponent, y : exponent;}\\&\text{exp(exp(g, x), y) = exp(exp(g, y), x).}\end{aligned}$$

Security Goals Modeling: Here, we provide a brief description of how the security requirements mentioned in Section IV-C are modeled.

Authentication: ProVerif provides correspondence assertions to capture the authentication or relationship between parties or events. We use both basic and injective correspondence assertions to capture non-injective (which also covers weak agreement) and injective agreements between parties, respectively. If the specified events occur in the correct sequence and the parameters remain consistent, the corresponding attribute is validated.

The following query demonstrates that if party A executes event $e1$ with parameter x, then party also performs event $e2$ using the same parameter. This implies that, from party A's perspective, party B has achieved non-injective correspondence with A concerning parameter x. An example of such non-injective correspondence is shown below:

$$\begin{aligned}&\text{query x : bitstring, t1, t2 : time;}\\&\text{event (e1(x)@t1)==> (event(e2(x))@t2 \&\& t1 > t2).}\end{aligned}$$

The inclusion of the temporal parameters $t1, t2$ further refines the query by associating each event with its occurrence time, thereby enabling precise tracking of causality and potential attack vectors. Similarly, an example of injective correspondence is shown below:

$$\begin{aligned}&\text{query x : bitstring, t1, t2, t3, t4, t5 : time;}\\&\text{inj − event (e1(x)@t1)==>}\\&\text{(inj − event(e2(x))@t2 \&\& t1 > t2).}\end{aligned}$$

This injective correspondence implies that there is a one-to-one relationship between the number of protocol runs performed by each participant. The injective correspondence assertion asserts that for each occurrence of event $e1(x)$, there is a distinct earlier occurrence of event $e2(x)$.

Secrecy: We leverage the reachability property of ProVerif to prove our secrecy claims. The reachability property enables the ProVerif tool to automatically search for any terms accessible to an attacker. Therefore, if a term, e.g., x, is accessible to an attacker, ProVerif can help identify the attack vector. We use the following queries to check whether a term x is accessible to the attacker, where the term x could be the anchor key $\mathsf{K_{SEAF}}$, long-term secret key k, subscriber identity SUPI, and HN's private key $\mathsf{sk_{HN}}$. For instance, the following query allows ProVerif to verify whether x remains confidential:

$$\begin{aligned}&\text{query x : bitstring, t1, t2, t3, t4, t5 : time;}\\&\text{((event(e1(x))@t1 \&\& attacker(x)@t2) ==> false).}\end{aligned}$$

Modeling Active Attacker and Subscribers Privacy: In ProVerif, achieving the indistinguishability property entails ensuring that an active attacker cannot distinguish two different processes in the protocol execution. This is typically accomplished by designing the protocol in a manner that prevents any information leaked to the active attacker from providing an advantage in distinguishing between these processes. The concept of indistinguishability is represented using observational equivalence in ProVerif.

To model observational equivalence between two processes in our model, we utilize the construct $\mathsf{Choice[M, M']}$, which provides a single *biprocess* encoding both processes. The $\mathsf{Choice[M, M']}$ encapsulates the terms that differ between the two processes: one process uses the first component, M, while the other process uses the second one, $\mathsf{M'}$. Our analysis reveals that these two processes exhibit equivalence, implying they possess identical structures and differ only in the selection of terms. This finding suggests that an active attacker cannot distinguish between these distinct processes during protocol execution. Hence, both of our protocols support protection against active attackers.

B. Formal Verification Results

Table II summarizes the security requirements supported by our protocols using ProVerif, as outlined in Section IV. Specifically, both protocols achieve injective agreement on $\mathsf{K_{SEAF}}$ for each pair of parties. Furthermore, they achieve non-injective agreement on $\mathsf{ID_{SN}}$ between the subscriber UE and HN, while the SN achieves non-injective agreement on SUPI with the HN. Both protocols also ensure the secrecy of the anchor key $\mathsf{K_{SEAF}}$ and subscriber's identity SUPI. Additionally, Protocol II supports PFS. Our security analysis further demonstrates that both protocols effectively protect against active attackers (indistinguishability).

VII. Performance Analysis

In this section, we present a detailed comparison of our protocols with the 3GPP standardized 5G-AKA protocol [1] and its improved version 5G-AKA′ (USENIX'21) [13]. We start with a theoretical comparison, followed by experimental results.

A. Theoretical Comparison

This section compares theoretically our protocols with the 3GPP standardized 5G-AKA and 5G-AKA′.

Table III shows a comparison between our protocols with others in terms of the number of messages exchanged between entities in each protocol. Since neither of our protocols uses sequence numbers, they do not require a sequence number resynchronization phase. As a result, our protocols can be completed with only seven messages exchanged among the subscriber, SN, and HN. In contrast, 5G-AKA and 5G-AKA′ require 13 and 9 messages, respectively, when sequence number synchronization is necessary and when it is not. It is evident that our protocols require fewer message exchanges between entities, thereby reducing overall communication overhead. It is important to note that, compared to the 5G-AKA protocol, our protocols introduce an additional MAC parameter, which is transmitted from the subscriber to the HN via the SN along with the SUCI. As detailed in Section V-B, this addition enhances our protocols by avoiding the inefficient sequence number-based replay attack prevention mechanism used in 5G-AKA and by making them more resistant to active attackers. Additionally, our protocols introduce the kcMAC parameter, which is sent from the subscriber to the HN via the SN along with RES^*. This parameter enables explicit key confirmation with both the SN and HN, a feature not present in the 5G-AKA protocol. Moreover, while the 5G-AKA protocol requires an additional parameter, CONC, sent by the HN to the subscriber via the SN, our protocols do not require such a parameter.

Table IV compares the computation costs between our protocols and 5G-AKA as well as 5G-AKA′. We evaluate the computation costs incurred by the subscriber, SN, and HN during the execution of each protocol. For 5G-AKA and 5G-AKA′, we consider three scenarios: successful authentication (Case 1), SYNC_FAILURE (Case 2), and MAC_FAILURE (Case 3). For our protocols, we focus on Case 1 and Case 3, as Case 2 is not applicable[6]. In Cases 2 and 3, the computation costs reflect the effort required by the protocols to reach the SYNC_FAILURE and MAC_FAILURE phases, respectively.

We denote T_h, T_m, T_{enc}, T_{xor}, T_{add}, T_{prf}, and T_{dec} as the time required to perform one hash/KDF/MAC operation, elliptic curve scalar multiplication, encryption, XOR, addition, random number generation, and decryption operation, respectively. As shown in Table IV, our protocols incur slightly higher computation costs for the subscriber, SN, and HN compared to the 5G-AKA protocol in Case 1. In Case 3, our protocols involve relatively more computation at both the subscriber and HN sides. However, as mentioned earlier, Case 3 is likely to occur less frequently.

B. Experimental Results

We implemented both of our protocols as well as 5G-AKA and 5G-AKA′ to compare their computation times at the subscriber, SN, and HN sides. The simulations were conducted on a GPU Laptop 11 Enterprise (64-bit) machine with a 2.8 GHz Intel (R) Xeon(R) E-2276M and 32 GB of memory, using Microsoft Visual Studio 2022. We utilized the HTTPLIB library [34] to manage HTTP/HTTPS communications between the entities (i.e., subscriber, SN, and HN). All protocols were evaluated under the same security level. All our codes are available in https://anonymous.4open.science/r/AKA-5G-E8B4/.

We used the Crypto++ cryptographic library [17] to implement ECIES with the SECP256R1 curve. The Encryptor.Encrypt() and Decryptor.Decrypt() interfaces were modified to support the export and import of shared keys derived by ECIES. Additionally, we employed SHA256 with different prefixes as $\mathrm{f1}, \mathrm{f2}, \mathrm{f3}, \mathrm{f4}, \mathrm{f5}, \mathrm{f1}^*, \mathrm{f5}^*$. For key derivation and message authentication, we used Password-Based Key Derivation Function 2 (PBKDF2) and Hash-based Message Authentication Code (HMAC) for KDF and HMAC, respectively. The computation time is measured using the CHRONO library provided by C++.

Table V presents the actual computation times for the subscriber, SN, and HN to complete their respective operations, measured in microseconds. The results indicate that, in Case 1, our protocol I demonstrates a highly competitive performance, with a computation cost only 0.05% higher than the 5G-AKA protocol and 2.29% lower than the 5G-AKA′ protocol at the subscriber side. Our protocol II also performs well, with just 0.13% additional computation time compared to the 5G-AKA protocol and 2.21% less compared to the 5G-AKA′ protocol on the subscriber side.

Both of our protocols require an additional hash operation at the SN compared to the 5G-AKA and 5G-AKA′ protocols, resulting in a 48% increase in computation time. While the 5G-AKA and 5G-AKA′ protocols involve a single hash operation at the SN, our protocols incorporate two hash operations. This enhancement significantly boosts the security of our protocols, justifying the increased computation time. At the HN side,

[6]Our protocols do not require the sequence number resynchronization phase.

our protocol I requires 5.45% and 2.67% more computation time than the 5G-AKA and 5G-AKA′ protocols, respectively. Our protocol II demonstrates a slightly higher increase, with 20.91% and 17.71% more computation time compared to the 5G-AKA and 5G-AKA′ protocols, respectively, reflecting its enhanced security benefits.

Our protocols avoid Case 2, which also provides better overall efficiency and reduces the computational overhead associated with sequence number resynchronization. This contributes to a more streamlined and effective authentication process, enhancing performance and reliability.

Case 3 occurs less frequently, typically due to an active attacker or a misconfiguration in the network or HN. Nevertheless, we evaluate the performance of our protocols under this scenario. At the subscriber side, our protocols incur approximately 55.02% more computation time compared to the 5G-AKA protocol, and about 50% more than the 5G-AKA′ protocol. On the HN side, Protocol I and Protocol II introduce increases of approximately 2.8% and 17%, respectively, over the 5G-AKA protocol. Compared to the 5G-AKA′ protocol, Protocol I requires 2.5% more computation time, while Protocol II requires 16.7% more. Given the rarity of this scenario, the additional computational overhead introduced by our protocols is considered acceptable, especially in light of the enhanced security guarantees they provide.

VIII. Conclusion

In this paper, we proposed AKA protocols for 5G that achieve all security goals specified in the 3GPP technical specification TS 33.501, along with important underspecified security objectives. We designed two protocols: the first provides all security guarantees except perfect forward secrecy, and its extended version also supports perfect forward secrecy with minimal computational overhead. Both protocols are compatible with existing SIM cards, resist both passive and active attacks, and may require only software modifications on the subscriber, SN, and HN. We verified the claimed security goals using ProVerif. Our implementation results and comparisons with the existing 5G-AKA and 5G-AKA′ protocols demonstrate that our protocols offer enhanced security while providing better or comparable computation and communication overhead, making them well-suited for 5G and beyond.

While our simulation-based evaluation provides a comparative analysis of computational and communication overhead, it does not fully capture the operational dynamics of real-world 5G environments. Factors such as device heterogeneity, network variability, and mobility were not modeled. Additionally, integration with actual mobile network stacks and adversarial testing in live settings remains an open area for future work. We plan to extend our evaluation using real-world testbeds such as OpenAirInterface (OAI) to validate the practical effectiveness and robustness of our protocols under realistic conditions. Furthermore, while our protocol is designed to be compatible with existing USIM cards based on current 3GPP specifications, we acknowledge that this compatibility has not yet been experimentally verified. Additional testing is needed to confirm practical interoperability, which remains part of our future work.

Acknowledgment

This research paper is conducted under the 6G Security Research and Development Project, as led by the Commonwealth Scientific and Industrial Research Organisation (CSIRO) through funding appropriated by the Australian Government's Department of Home Affairs. This paper does not reflect any Australian Government policy position. For more information regarding this Project, please refer to https://research.csiro.au/6gsecurity/.

References

[1] 3GPP. TS 33.501: Security Architecture and Procedures for 5G System-v18.5.0. 2024, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169, [Accessed: 4-April-2024].

[2] D. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler. A Formal Analysis of 5G Authentication. In *Proc. of the 2018 ACM CCS*, page 1383–1396, 2018.

[3] C. Cremers and M. Dehnel-Wild. Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion. In *Proc. of the NDSS*, 2019.

[4] R. Miller, I. Boureanu, S. Wesemeyer, and C. J. P. Newton. The 5G Key-Establishment Stack: In-Depth Formal Verification and Experimentation. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '22, page 237–251, 2022.

[5] M. Arapinis *et al.* New privacy issues in mobile telephony: fix and verification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, page 205–216, 2012.

[6] M. S. A. Khan and C. J. Mitchell. "improving air interface user privacy in mobile telephony". In *Security Standardisation Research*, pages 165–184, Cham, 2015. Springer International Publishing.

[7] R. Borgaonkar, L. Hirschi, S. Park, and A. Shaik. New Privacy Threat on 3G, 4G, and Upcoming 5G AKA Protocols. *Proc. of PETS*, 2019(3):108–127, 2019.

[8] A. Koutsos. The 5G-AKA Authentication Protocol Privacy. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 464–479, 2019.

[9] P. A. Fouque, C. Onete, and B. Richard. Achieving Better Privacy for the 3GPP AKA Protocol. In *Proceedings on PoPETs'16*, page 255–275, 2016.

[10] Free 5GC - Link the World, https://www.free5gc.org/ [Accessed: 4th Sept. 2024].

[11] Open5gcore - the next mobile core network testbed plat- form, https://www.open5gcore.org/ [Online Access: 4th Sept. 2024].

[12] Open5gcore - the next mobile core network testbed plat- form, https://www.openairinterface.org/ [Online Access: 4th Sept. 2024].

[13] Y. Wang, Z. Zhang, and Y. Xie. Privacy-Preserving and Standard-Compatible AKA Protocol for 5G. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3595–3612. USENIX Association, Aug. 2021.

[14] A. Braeken. Symmetric key based 5G AKA authentication protocol satisfying anonymity and unlinkability. *Computer Networks*, 181:107424, 2020.

[15] J. Munilla, M. Burmester, and R. Barco. An enhanced symmetric-key based 5G-AKA protocol. *Computer Networks*, 198:108373, 2021.

[16] B. Blanchet. *"Automatic Verification of Security Protocols in the Symbolic Model: The Verifier ProVerif"*, pages 54–87. 2014.

[17] https://www.cryptopp.com/ [Online Access: 4th Sept. 2024].

[18] M. Khan, P. Ginzboorg, K. Järvinen, and V. Niemi. "defeating the downgrade attack on identity privacy in 5g". In *Security Standardisation Research*, pages 95–119, Cham, 2018. Springer International Publishing.

[19] F. van den Broek, R. Verdult, and J. de Ruiter. Defeating IMSI Catchers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 340–351, 2015.

[20] J. Arkko, K. Norrman, M. Näslund, and B. Sahlin. A USIM Compatible 5G AKA Protocol with Perfect Forward Secrecy. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 1205–1209, 2015.
[21] F. Liu, J. Peng, and M. Zuo. Toward a Secure Access to 5G Network. In *2018 17th IEEE TrustCom/BigDataSE*, pages 1121–1128, 2018.
[22] I. You *et al.* 5G-AKA-FS: A 5G Authentication and Key Agreement Protocol for Forward Secrecy. *Sensors*, 24(1), 2024.
[23] M. T. Damir *et al.* A Beyond-5G Authentication and Key Agreement Protocol. In *Network and System Security*, pages 249–264, 2022.
[24] Tamarin (develop). https://github.com/tamarin-prover/tamarin-prover, [Accessed: 14- Aug.- 2024].
[25] 3GPP. TR 33.902: Formal Analysis of the 3G Authentication Protocol (Release 4), Sept. 2001. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2337, [Accessed: 14- Aug.- 2024].
[26] 3rd Generation Partnership Project (3GPP). 3GPP TS 31.121: UICC-terminal interface; Universal Subscriber Identity Module (USIM) application test specification. https://www.3gpp.org/DynaReport/31121.htm, 2023. Version 17.6.0.
[27] J. Zhao, B. Ding, Y. Guo, Z. Tan, and S. Lu. Securesim: rethinking authentication and access control for sim/esim. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, MobiCom '21, page 451–464, 2021.
[28] T. P. Lisowski, M. Chlosta, J. Wang, and M. Muench. SIMurai: Slicing through the complexity of SIM card security research. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4481–4498. USENIX Association, Aug. 2024.
[29] 3rd Generation Partnership Project (3GPP). 3GPP TS 33.105: 3G Security; Cryptographic algorithm requirements. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2264, 2024. Version 18.0.0.
[30] V. Shoup. A Proposal for an ISO Standard for Public Key Encryption. Cryptology ePrint Archive, Paper 2001/112, 2001. https://eprint.iacr.org/2001/112.
[31] G. Lowe. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*, pages 31–43, 1997.
[32] ProVerif 2.05. https://bblanche.gitlabpages.inria.fr/proverif/.
[33] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.
[34] https://github.com/yhirose/cpp-httplib [Online Access: 4th Sept. 2024].

APPENDIX A
ELLIPTIC CURVE INTEGRATED ENCRYPTION SCHEME (ECIES) [30]

5G uses ECIES cryptographic primitive to protect the unique identity of the subscribers (please refer to 3GPP TS 33.501 [1]). ECIES is a hybrid encryption scheme based on public key cryptography, comprising a Key Encapsulation Mechanism (KEM) and a Data Encapsulation Mechanism (DEM). In ECIES, the KEM is used to establish shared keys between the sender and recipient through public key cryptography. Subsequently, a DEM encrypts and decrypts the actual payload using symmetric cryptography with the shared key. The ECIES consists of the following algorithms.

KEYGEN $((\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{pp})$*:* It takes an elliptic curve domain parameters pp as input and outputs a private key sk and a public key pk, where $\mathsf{sk} \in \mathbb{Z}_q^*$ (a multiplicative group of integer modulo q), $\mathsf{pk} = \mathsf{sk} \cdot g$, and g is a generator of the chosen elliptic curve. 3GPP recommends two elliptic curves CURVE25519 and SECP256R1 [1].

ENCAP$((C_0, \mathsf{k_s}) \leftarrow \mathsf{pk})$*:* It takes the public key pk as input to generate an ephemeral private-public key pair (r, R), where $r \in \mathbb{Z}_q^*$ and $R = r \cdot g$. It sets $C_0 = R$ and computes a shared secret key $\mathsf{k_s}$, where $\mathsf{k_s} = \mathrm{KDF}(r \cdot \mathsf{pk})$.

DECAP $(\mathsf{k_s} \leftarrow (C_0, \mathsf{sk}))$*:* It takes C_0 and the private key sk as input and outputs the shared secret key $\mathsf{k_s}$ as follows $\mathsf{k_s} = \mathrm{KDF}(\mathsf{sk} \cdot C_0)$.

SENC $((C_1, C_2) \leftarrow (\mathsf{k_s}, M))$*:* It takes the shared secret key $\mathsf{k_s}$ and a message M as input. It outputs two ciphertext components (C_1, C_2), where $C_1 = \textsc{Enc}(\mathsf{s_1}, M)$ is the encrypted component of the message M using a symmetric key encryption algorithm, and $C_2 = \mathrm{MAC}(\mathsf{s_2}, C_1)$ is a message authentication code to check the integrity and authenticity of C_1. Here, $(\mathsf{s_1}, \mathsf{s_2})$ are the leftmost and rightmost octets of the shared secret key $\mathsf{k_s}$.

SDEC $(M \leftarrow (\mathsf{k_s}, C_1, C_2))$*:* It takes the shared secret key $\mathsf{k_s}$, C_1, and C_2 as input. It outputs the actual message M as follows: It extracts $(\mathsf{s_1}, \mathsf{s_2})$ from $\mathsf{k_s}$, verifies if $C_2 = \mathrm{MAC}(\mathsf{s_2}, C_1)$, and if the verification is successful, decrypts C_1 to obtain M using $\textsc{Dec}(\mathsf{s_1}, C_1)$.

Please refer to [30] for more details on the ECIES algorithms.

APPENDIX B
DETAILED DESCRIPTION OF 5G-AKA PROTOCOL

We provide a detailed description of the 5G-AKA protocol here. We start with the *Initiation* phase, followed by the *Challenge-Response*, *Sequence Number Re-synchronization*, and *MAC Failure* phases. Figure 1 shows a high-level overview of the 5G-AKA protocol.

Initiation: This phase starts once the session between the subscriber and SN is initialized. The subscriber encrypts its unique identity SUPI with the HN's public key $\mathsf{PK_{HN}}$ using ECIES (please refer to Appendix A for more details on ECIES) and produces a ciphertext SUCI. The subscriber then sends SUCI and the identity of the HN, $\mathsf{ID_{HN}}$ to the SN. Afterward, the SN forwards the received SUCI to the HN, adding its own identity $\mathsf{ID_{SN}}$. Upon receiving SUCI, the HN decrypts it using its own private key $\mathsf{sk_{HN}}$ and retrieves SUPI, which helps the HN obtain the long-term secret key k and sequence number $\mathsf{SQN_{HN}}$ associated with the subscriber's SUPI from its database.

Challenge-Response: In this phase, the subscriber and the HN mutually authenticate each other using a challenge-response method. Additionally, the subscriber and the SN establish the anchor keys, $\mathsf{K_{SEAF}}$, for any further secure communication.

The HN chooses a random challenge R and generates a tuple $\langle R, \mathrm{AUTN} = \langle \mathrm{CONC}, \mathrm{MAC} \rangle, \mathrm{HXRES}, \mathsf{K_{SEAF}} \rangle$, which is then sent to the subscriber. In CONC, the HN conceals the sequence number associated with the subscriber $\mathrm{SQN_{HN}}$ using the anonymous key AK derived from R and k. The MAC is computed for the authentication and integrity checking of the random challenge R, using R, k, and $\mathrm{SQN_{HN}}$. Additionally, HXRES is computed by hashing R with the expected response XRES from the subscriber, which is computed using R and k. Finally, the anchor keys $\mathsf{K_{SEAF}}$ are computed using k, R, $\mathrm{ID_{SN}}$, and $\mathrm{SQN_{HN}}$. The HN also increments the sequence number $\mathrm{SQN_{HN}}$ by 1 at the end.

After receiving the tuple $\langle R, \text{AUTN}, \text{HXRES}, \mathsf{K_{SEAF}} \rangle$, the SN stores a copy of $\langle R, \text{HXRES}, \mathsf{K_{SEAF}} \rangle$ and forwards the tuple $\langle R, \text{AUTN} \rangle$ to the subscriber.

Upon receiving the tuple $\langle R, \text{AUTN} \rangle$, the subscriber computes an anonymous key AK inside the USIM card using k and R, and uncovers the sequence number SQN_{HN} from the CONC part of AUTN. Next, it checks MAC using R and SQN_{HN} inside the USIM card. If this check fails, it returns a MAC_FAILURE message, which is sent to the SN and goes to the *MAC Failure* phase. If the check is successful, the subscriber verifies the freshness of SQN_{HN}. If this verification fails, the USIM card returns $\text{AUTN} = \langle \text{CONC}, \text{MAC}^* \rangle$, where CONC conceals the sequence number SQN_{UE}, and sends the tuple $\langle$SYNC_FAILURE$, \text{AUTN} \rangle$ to the HN to synchronize the sequence number. We shall present the re-synchronization process later. If the freshness check is successful, the USIM card sets SQN_{UE} to SQN_{HN}, computes the anchor keys $\mathsf{K_{SEAF}}$ using k, R, ID_{SN}, and SQN_{HN}, and finally returns the tuple $\langle \mathsf{K_{SEAF}}, \text{RES} \rangle$. The subscriber keeps $\mathsf{K_{SEAF}}$ and sends RES to the SN.

After receiving RES, the SN compares the hash of RES and R with HXRES. If successful, the SN forwards RES to the HN, which then compares RES with its stored XRES. If this comparison is successful and the subscriber is authenticated, the HN sends the SUPI associated with the subscriber to the SN. This concludes the 5G-AKA protocol execution for the current session.

Please note that 3GPP TS 33.501 [1] also specifies that the subscriber and SN should implicitly confirm the agreed keys and each other's identities through the successful use of keys in subsequent procedures. This can be achieved with an additional key-confirmation round trip using $\mathsf{K_{SEAF}}$.

Sequence Number Re-synchronization: This phase is initiated when the subscriber needs to re-synchronize its sequence number with the HN. The primary purpose of using a sequence number-based freshness check is to prevent replay attacks. However, factors such as message loss or system failure may lead to a desynchronization between the subscriber's SQN_{UE} and the HN's SQN_{HN}.

When the freshness check fails, as previously mentioned, the USIM card returns $\text{AUTN} = \langle \text{CONC}, \text{MAC}^* \rangle$, with CONC concealing the sequence number SQN_{UE}. It then sends the tuple $\langle$SYNC_FAILURE$, \text{AUTN} \rangle$ to the HN to synchronize the sequence number. Upon receiving the Sync_Failure message, the SN forwards the tuple $\langle$SYNC_FAILURE$, \text{AUTN}, R, \text{SUCI} \rangle$ to the HN. Subsequently, the HN de-conceals the SQN_{UE} after verifying the message authentication code MAC^*. If the verification is successful, the HN sets its sequence number SQN_{HN} to $\text{SQN}_{\text{UE}} + 1$.

MAC Failure: This phase is initiated when the MAC check fails in the *Challenge-Response* phase. In this phase, the subscriber simply returns a $\text{MAC}_{\text{F}}\text{ailure}$ message and goes to the *Initiation* phase to restart the AKA protocol in a new session.

APPENDIX C
PRIVACY ISSUES WITH 5G-AKA PROTOCOL

In this section, we briefly discuss the three main types of privacy-related attacks that can take place in the 5G-AKA protocol. Please note that these three types of attack are well explained in [13]. The three types of privacy-related attacks the 5G-AKA protocol is vulnerable to are: *Failure Message Linkability Attack* [5], *Sequence Number Inference Attack* [7], and *Encrypted SUPI Replay Attack* [8], [9].

Failure Message Linkability Attack: The goal of this attack is to distinguish a targeted subscriber from others by analyzing the responses received after replaying records of $\langle R, \text{AUTN} \rangle$ to all subscribers in the vicinity. The targeted subscriber responds with a SYNC_FAILURE message because the replayed message passes the initial MAC verification with the correct long-term secret key k but fails the freshness check SQN_{HN}. In contrast, other subscribers respond with a MAC_FAILURE message, as the replayed message fails the MAC verification due to a mismatched long-term secret key k.

Sequence Number Inference Attack: The objective of this attack is to learn information about the targeted subscriber's sequence number SQN_{UE}. The attacker replays previously captured tuples $\langle R, \text{AUTN} \rangle$ multiple times and captures the returned CONC in the SYNC_FAILURE messages. The attacker attempts to learn $\text{SQN}^{\text{i}}_{\text{UE}} \oplus \text{SQN}^{\text{i+1}}_{\text{UE}}$ by performing an Exclusive-OR operation between CONC^{i} and $\text{CONC}^{\text{i+1}}$, as both CONC^{i} and $\text{CONC}^{\text{i+1}}$ have concealed their sequence numbers using the same anonymous key AK.

Encrypted SUPI Replay Attack: The goal of this attack is to distinguish the targeted subscriber from others. In this attack, the attacker replays the captured SUCI during the Initiation phase to the HN in all subscriber sessions and waits for the subscribers' responses to the corresponding challenge messages from the HN. The targeted subscriber responds successfully, while the other subscribers reply with MAC_FAILURE messages because the long-term secret key k used to generate the MAC matches only for the targeted subscriber and not for the others.